Guida alla progettazione e alla realizzazione dei siti web

di

Marco Andreini, Patrizia Andronico,
Arianna Del Soldato, Cristian Lucchesi,
Maurizio Martinelli, Giuseppe Vasarelli

Istituto di Informatica e Telematica del CNR

Indice

1 Introduzione 9

2 Il servizio WWW 15
2.1 Introduzione 17
2.2 Cosa è il World Wide Web 18
2.3 Le origini del web 20
2.4 La nascita del WWW 22
2.5 Storia del WWW 25
2.6 Il web ed internet 27
2.6.1 Le connessioni fisiche: cavi e strade 27
2.6.2 Le regole di comportamento comuni 27
2.6.3 Servizi per tutti 28
2.7 Come funziona il web 29
2.8 Principi fondamentali del WWW 31
2.8.1 Leggibilità universale 31
2.8.2 Gli ipertesti 31
2.8.3 La ricercabilità delle informazioni 32
2.8.4 Modello client-server 32
2.8.5 La negoziazione del formato 32
2.8.6 Le tecnologie ed i protocolli fondamentali del WWW 32
2.9 HTTP 34
2.10 Breve storia dell'HTTP 37
2.10.1 Cosa è una RFC? 37
2.11 HTTP 1.1 39
2.11.1 HTTP 1.1: Status Code più comuni 39
2.11.2 HTTP 1.1: categorie di Status Code 40
2.11.3 Richieste multiple 41
2.11.4 Connessione permanente 42
2.12 URI 43
2.12.1 Composizione delle URI 43
2.12.2 URI Schema 44
2.12.3 URI HTTP 45
2.12.4 Struttura ad albero degli URI http 45
2.12.5 Elementi riservati, e valori predefiniti nelle URI 46
2.12.6 URL 46
2.13 Linguaggi di markup 48
2.14 HTML 49
2.15 Non c'è solo HTML 50
2.16 Il W3C 51

3 Progettare un sito – parte 1 53
3.1 Capire i media 55
3.1.1 Cosa è un sito web 55
3.1.2 Comunicazione/informazione 59
3.1.3 Telecomunicazioni e memoria 61
3.2 Web design1 64

3.2.1 La raccolta di informazioni ... 64
3.2.2 Il cliente ... 65
3.2.3 L'utente ... 65
3.2.4 Gli scenari d'uso ... 66
3.2.5 Il re-design ... 66
3.2.6 Il project plan ... 67
3.3 Visual design ... 68
3.3.1 La lettura su web ... 68
3.3.2 Scelte tipografiche ... 70
3.3.3 Layout di pagina ... 73

4 Introduzione all'HTML ... 77
4.1 Cosa è l'HTML? ... 79
4.2 Struttura del documento ... 80
4.2.1 I tag ... 80
4.2.2 Gli elementi ... 80
4.2.3 Gli elementi in XHTML ... 81
4.2.4 Elementi vuoti ... 81
4.2.5 I caratteri di spaziatura ... 81
4.2.6 Gli attributi ... 82
4.2.7 Differenze tra HTML e XHTML ... 82
4.2.8 Annidamento ... 83
4.3 Codifiche dei caratteri ... 84
4.3.1 ASCII ... 84
4.3.2 ISO-8859 e altre codifiche ... 84
4.4 Codifica del documento ... 86
4.4.1 Caratteri speciali (entità) ... 87
4.4.2 Spazi che uniscono: nbsp ... 88

5 HTML: struttura di una pagina ... 89
5.1 Struttura del documento HTML ... 91
5.1.1 Definizione del tipo di documento ... 91
5.1.2 Il documento HTML ... 94
5.1.3 Intestazione del documento ed elementi contenuti ... 96
5.1.4 Corpo del documento ... 102

6 6 CSS - parte 1 ... 115
6.1 Cascading Style Sheet (CSS) ... 117
6.2 Composizione degli stylesheet ... 118
6.3 Raggruppare le regole ... 120
6.3.1 Raggruppare per dichiarazione ... 120
6.3.2 Raggruppare per selettore ... 121
6.3.3 Raggruppare per dichiarazione e per selettore ... 121
6.4 Le specifiche CSS ... 122

6.4.1 CSS 2.1 vs CSS 2 122
6.4.2 CSS 3: il futuro 123
6.5 Associare gli stylesheet all'HTML 124
6.5.1 L'attributo style 124
6.5.2 L'elemento style 125
6.5.3 L'elemento link 126
6.6 Gli Stylesheet esterni 127
6.7 Struttura ad albero ed ereditarietà 128
6.7.1 Sovrascrivere l'ereditarietà 129
6.7.2 Proprietà che non si ereditano 129
6.7.3 Proprietà che non si ereditano: background 130
6.8 Un utilizzo comune dei CSS: i font 132
6.8.1 font: "font-size" 132
6.8.2 Font-size specificato per altezza o in percentuale 132
6.8.3 Unità di misura 133
6.8.4 Unità di misura relative 133
6.8.5 Esempi di font-size per altezza 134
6.8.6 Font-size per valore assoluto 134
6.8.7 Font-size per valore relativo 134
6.8.8 Font: "font-weight" 135
6.8.9 Esempi di font-weight 135
6.8.10 Famiglie di font-family 136
6.8.11 Font specifici 136
6.8.12 Font: font-family 136
6.8.13 Indicare uno o più font 137
6.9 Altre proprietà impostabili con i CSS 138
6.9.1 La spaziatura all'interno degli elementi 138
6.9.2 Allineamento del testo: text-align 138
6.9.3 Indentazione del testo e altezza del righe: text-indent, line-height 139
6.9.4 Spazi tra parole e lettere: word spacing, letter-spacing 140
6.9.5 Allineamento verticale del testo: vertical-align 140
6.9.6 white-space 141
6.10 Perché i CSS? 142
6.10.1 Il (non) problema della retro compatibilità 142

7 Altri elementi HTML 143
7.1 Collegamenti ipertestuali 145
7.1.1 Attributo "href" e URI relative 148
7.1.2 Attributo "href" e protocolli 150
7.2 Tabelle 152
7.2.1 Definizione delle celle 153
7.2.2 Definizione del contenuto 154

7.2.3 Gli elementi colgroup e col 158

8 Le immagini 161
8.1 Inserire immagini 163
8.2 Formati di immagine 164
8.2.1 Graphics Interchange Format (GIF) 164
8.2.2 Joint Photographic Experts Group (JPEG) 166
8.2.3 Portable Network Graphics (PNG) 166
8.3 Quando usare immagini 168
8.4 Quando usare testo 169
8.5 Velocizzare lo scaricamento delle immagini 170
8.6 L'elemento `img` 171
8.6.1 L'attributo `src` 171
8.6.2 L'attributo `alt` 173
8.6.3 Gli attributi `height` e `width` 174
8.6.4 Gli attributi `ismap` e `usemap` 176
8.7 Le immagini sensibili al mouse 178
8.7.1 Immagini sensibili "server-side" 178
8.7.2 Immagini sensibili "client-side" 179
8.8 L'elemento `map` 180
8.9 L'elemento `area` 181
8.9.1 L'attributo `shape` 181
8.9.2 L'attributo `coords` 182
8.10 Le immagini di sfondo 185
8.11 I colori del documento 186
8.11.1 I valori dei colori 186
8.11.2 I nomi dei colori 187
8.11.3 La mappa standard 187
8.12 Gli editor di immagini 189

9 Struttura di una pagina HTML 191
9.1 Struttura della pagina HTML 193
9.1.1 Elementi permessi 193
9.1.2 Introduzione alla validazione 194
9.1.3 Elementi appropriati per il testo 194
9.1.4 Non solo tabelle 195
9.1.5 Formule su Web 195
9.1.6 Elementi deprecati 196

10 Progettare un sito – parte 2 197
10.1 Web design 2 199
10.1.1 Il team di sviluppo 199
10.1.2 Sviluppare la struttura del sito 201
10.2 UCD – User Centered Design 207
10.2.1 Progettazione Centrata sull'Utente 207

10.2.2 Principi di design di Norman 208
10.2.3 Il ciclo della UCD 212
10.3 Usabilità 214
10.3.1 Definizioni di usabilità 214
10.3.2 I principali problemi di usabilità 215
10.3.3 Leggi di Krug sull'usabilità 218
10.4 Come si misura l'usabilità 219
10.4.1 Valutazione euristica 220
10.4.2 test di usabilità 222

11 CSS – parte 2 223
11.1 CSS predefiniti! 225
11.1.1 CSS multipli 227
11.2 Ancora sui selettori 228
11.2.1 Le classi e l'attributo CLASS 228
11.2.2 Il selettore per ID e l'attributo ID 228
11.2.3 Combinare i selettori: tipo e classe 229
11.2.4 Usare più classi 229
11.2.5 Combinare i selettori per contesto 230
11.2.6 Selettori: * 232
11.3 Pseudo classi 233
11.3.1 Pseudo Classi: hover, active, focus 233
11.3.2 Link, visited 234
11.3.3 Selezione per contesto: first-child 234
11.3.4 First-letter, first-line 235
11.3.5 Selettori: una tabella riassuntiva 235
11.4 Box Model 237
11.4.1 I tipi di box e la proprietà Display 237
11.4.2 Display: block 238
11.4.3 Display: inline 238
11.4.4 Display:none 238
11.4.5 Display:list-item 239
11.5 Personalizzare le liste 240
11.5.1 Personalizzare le liste: list-style-type 240
11.5.2 List-style-image 241
11.5.3 List-style-position 241
11.5.4 List-style 241
11.6 I colori: la proprietà color 243
11.7 Background 244
11.7.1 background-image, background-repeat 244
11.7.2 background-position 244
11.7.3 Proprietà background 245

11.8 Box Model: dimensioni....246
11.9 Un utilizzo comune dei CSS: i margini, il padding ed i bordi....247
11.9.1 I margini....247
11.9.2 Proprietà margin....248
11.9.3 padding....248
11.9.4 I Bordi....249
11.9.5 border-width....249
11.9.6 border-color....250
11.9.7 border-style....250
11.9.8 Proprietà border....251
11.10 width, height....252
11.10.1 width, margin-left, margin-right: auto....252
11.11 Schemi di posizionamento nei CSS....254
11.11.1 Posizionamento degli elementi....254
11.11.2 Posizionamento relativo....254
11.11.3 Posizionamento fisso....256
11.11.4 Posizionamento assoluto....256
11.12 Tabelle....260
11.13 Gli altri media....262

12 Cosa si fa e cosa si dovrebbe fare....263
12.1 Ancora sulle tabelle....265
12.1.1 Quando non usare le tabelle....265
12.2 Costruire il layout con stylesheet....266
12.2.1 Menu in CSS....266
12.2.2 Posizionamento e layout con i CSS....268

13 Progettare un sito – parte 3....271
13.1 Cosa è l'accessibilità....273
13.1.1 L'ISO e l'accessibilità....274
13.1.2 Il W3C e l'accessibilità del web....275
13.1.3 Usabilità e accessibilità....276
13.2 Cosa sono le WCAG....277
13.2.1 La WCAG Recommendation 1.0....278
13.2.2 La WCAG Recommendation 2.0....279
13.2.3 Differenze fra WCAG 1.0 e WCAG 2.0....281
13.3 Legislazione di riferimento....282
13.3.1 La Rehabilitation Act Amendments (Section 508)....282
13.3.2 La Commissione Europea....282
13.3.3 La situazione italiana....283
13.3.4 La legge Stanca....285
13.4 Valutare l'accessibilità....287
13.4.1 I validatori automatici....287
13.4.2 Dichiarazione di conformità....289

14 Interagire con il Web 291
14.1 Interagire col web 293
14.1.1 Interazioni col web: le form 293
14.1.2 Action 294
14.1.3 Method 294
14.1.4 Enctype 295
14.1.5 Interazioni con le form via HTTP 295
14.2 Gli elementi delle form 297
14.2.1 L'elemento input 297
14.2.2 L'elemento textarea 299
14.2.3 L'elemento select 299
14.2.4 Come etichettare i campi 301
14.2.5 Come raggruppare i campi 301
14.3 Presentazione delle form 302
14.3.1 Alcuni eventi 302
14.3.2 Javascript 302
14.4 Approfondimenti 305
14.4.1 Perché seguire gli standard 305
14.4.2 Cenni sull'accessibilità 305
14.4.3 Futuro degli ipertesti su web 306
14.4.4 Metadati e indicizzazione 306
14.4.5 Statistiche del sito 307

15 Appendice A: ALBERO DEGLI ELEMENTI XHTML 1.1 309
Albero degli elementi HTML fino al terzo livello 311
15.1.1 Sottoalberi degli elementi 315

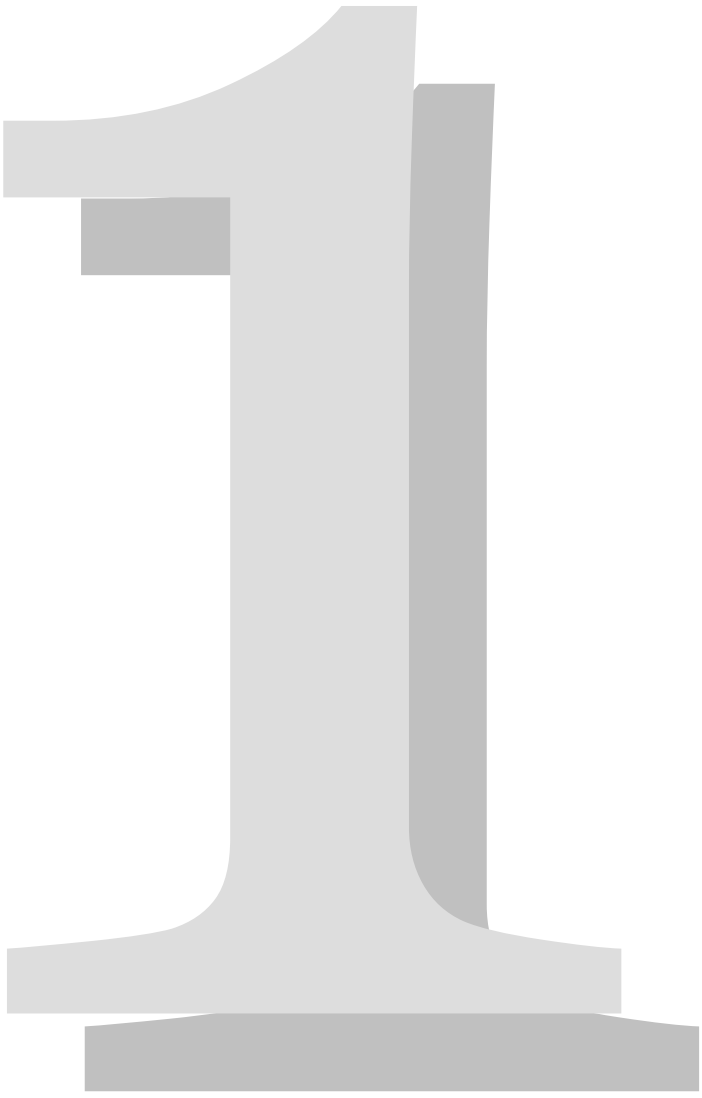
16 Appendice B: CARATTERI SPECIALI 348
16.1 Caratteri Latin-1 350
16.2 Caratteri Speciali 352
16.3 Symbols 353
16.3.1 Greek 353
16.3.2 General Punctuation 354
16.3.3 Letterlike Symbols 354
16.3.4 Arrows 354
16.3.5 Mathematical Operators 354
16.3.6 Geometric Shapes 355
16.3.7 Miscellaneous Symbols 355

17 appendice C: proprietà CSS 356
17.1 Proprietà CSS 358

18 Appendice D: COLORI 368
18.1 Colori 370

19 Bibliografia generale..... 376

Introduzione



Gli anni ’80 e i primi anni ’90 hanno visto proliferare tecnologie e strumenti volti a pubblicare e reperire informazioni sulla rete Internet. Sono nati in quegli anni i cosiddetti *Directory Services*, ossia servizi di informazione, tipo pagine bianche e gialle, della rete Internet, tra i quali i più rilevanti sono stati: il protocollo **Whois** (1982), il cui scopo iniziale era quello di memorizzare informazioni sulle reti e sulle persone coinvolte nella loro gestione; il protocollo **DNS** (Domain Name System 1983-1987) il cui funzionamento è tuttora alla base del meccanismo di risoluzione dei nomi a dominio e degli indirizzi IP; il protocollo **X.500** (1987) e il suo successore, tuttora in uso, **LDAP** (Lightweight Directory Access Protocol) con l’obiettivo iniziale di creare una sorta di elenco telefonico mondiale degli utenti Internet. Ma se da un lato si pensava a progettare e realizzare strumenti per la pubblicazione in rete delle informazioni, dall’altro si rendevano sempre più indispensabili strumenti e tecnologie atte a reperire tali informazioni, senza dover necessariamente essere a conoscenza di dove esse erano pubblicate. Nacquero così i primi sistemi di *Information Retrieval*, quali ad esempio il **WAIS** (Wide Area Information Servers) della Thinking Machines Corporation, un sistema informativo distribuito al quale era possibile accedere con opportuni programmi client; l’**Hytelnet** (1990), il primo directory Internet ipertestuale di servizi accessibili via telnet; il **Netfind** (1991), strumento rivoluzionario e ingegnoso sviluppato presso l’Università del Colorado che cercava di approcciare il problema del recupero dei dati utilizzando anche tecniche di *Information Discovery*. Sfruttando le potenzialità e la sinergia di protocolli e servizi di informazione eterogenei (posta elettronica, dns, finger, telnet, whois), Netfind era in grado di reperire le informazioni nei modi più disparati. Ma il primo vero sistema di *information retrieval* di successo è stato il **Gopher**, sviluppato dalla Università del Minnesota nel 1991: si trattava di un sistema, utilizzabile su terminali a caratteri, nel quale i dati erano organizzati secondo una struttura gerarchica e potevano contenere collegamenti (link) ad altri siti. Tra le innovazioni più rilevanti, rispetto ai suoi predecessori, Gopher introduceva la possibilità di rendere disponibili agli utenti informazioni quali suoni e immagini.

Ma la vera rivoluzione Internet è stata innescata dalla nascita del **World Wide Web** o **WWW**. Nel 1990, Tim Berners Lee, un ricercatore del CERN di Ginevra (Centro Europeo per la Ricerca Nucleare – il centro di ricerca per la fisica delle particelle più grande del mondo), presentava ai dirigenti dei laboratori una relazione intitolata "Information Management: a Proposal". La proposta, combinando le tecniche degli ipertesti e dell'*information retrieval*, aveva l'obiettivo di sviluppare un sistema di pubblicazione e reperimento dell'informazione distribuita su rete geografica eterogenea che permettesse alla comunità internazionale dei fisici, di condividere documenti, dati e, in generale, informazioni di qualsiasi natura. L'idea si rivelò originale e degna di essere presa in considerazione: organizzando l'informazione come una *ragnatela* di ipertesti e sfruttando sia i protocolli di comunicazione già esistenti che quelli sviluppati appositamente per accedere al World Wide Web, la comunità Internet sarebbe stata in grado, nel giro di pochi anni, di avere accesso, facilmente, alle miriadi di informazioni disponibili online. E fu proprio questa la forza del modello WWW: la capacità di interagire con la maggior parte dei sistemi di informazione esistenti tramite i loro stessi protocolli (o tramite appositi gateway). L'interazione client-server nella struttura WWW doveva avvenire, invece, tramite un nuovo protocollo chiamato *HyperText Transfer Protocol* (HTTP).

Nonostante la lungimiranza dell'idea e del nuovo modello proposto, il WWW ricevette un impulso decisivo al suo sviluppo solo nel 1993, quando il National Center for Supercomputing Applications (NCSA) dell'Università dell'Illinois, sviluppò l'interfaccia grafica multi piattaforma **Mosaic** per l'accesso ai documenti presenti sul Web. Da quel momento il numero di server Web nel mondo è cresciuto in maniera incredibile: nel 1993 esistevano al mondo solo 200 server Web, nel 1999 essi erano saliti a 7.800.000, a marzo 2006 essi sono circa 78.000.000!

A ulteriore testimonianza del successo e dell'importanza che la rete Internet e, di conseguenza il Web hanno avuto negli anni e di come tali tecnologie siano entrate a far parte della vita quotidiana delle persone, è sufficiente comparare le stime dei tassi di crescita della rete con quelli dei media

“tradizionali”: secondi gli analisti, per raggiungere la massa critica di 50 milioni di utenti sono occorsi 38 anni alla radio, 13 anni alla televisione e soltanto 5 a Internet!”.

Questa pubblicazione intende essere una guida semplice, pratica e snella a beneficio sia di coloro che da poco si sono avvicinati alla rete Internet e alle tecnologie Web, sia degli utente più “smaliziati” che sentono comunque la necessità di approfondire alcuni argomenti ancora oscuri o che comunque abbiano l’esigenza di disporre di un manuale veloce da consultare in caso di necessità. Gli autori hanno cercato di offrire un quadro esaustivo delle tematiche inerenti le tecnologie Web facendo tesoro della pluriennale esperienza maturata in questo campo. I lettori troveranno soprattutto soluzioni pratiche alle problematiche più frequenti. E, a nostro avviso, la praticità è proprio la forza di questa guida: pur garantendo la completezza dell’informazione, ogni argomento è stato affrontato in chiave concreta. L’essenziale, e nulla più.

Il servizio WWW

2.1 Introduzione

If I have seen farther than others, it is because I stood on the shoulders of giants. [Isaac Newton]

In questo capitolo verranno introdotti i concetti che stanno alla base delle tecnologie web. Nella prima parte del capitolo si descrive cosa si intende per World Wide Web per passare poi a riportare un pò di storia che permette di capire i passi fondamentali che hanno portato allo sviluppo del web così come lo conosciamo oggi. Il capitolo prosegue poi con la spiegazione del funzionamento del web e con l'introduzione dei concetti, protocolli e tecnologie che ne rendono possibile il suo funzionamento. Nella parte finale si parla brevemente del protocollo HTTP, del linguaggio HTML e si introduce il concetto di URI. Infine il capitolo si chiude con alcuni riferimenti alle tecnologie XML ed al ruolo del W3C nello sviluppo del web.

2.2 Cosa è il World Wide Web

Trovare una definizione unica che permetta di descrivere il World Wide Web (chiamato anche WWW o Web) è un compito molto difficile, sono state create molte definizioni nel tempo che prendono in considerazione alcuni aspetti del web ma nessuna è esaustiva, una definizione abbastanza completa è:

- il WWW è l'iniziativa che mira a fornire un "wide-area hypermedia information retrieval" (un sistema di reperimento di informazioni su larga scala) per l'accesso ad un vasto insieme di documenti.

Nonostante il WWW sia cambiato molto dal momento della sua invenzione la definizione data nel 1991 da Tim Berners-Lee (l'inventore del Web) è tuttora attuale:

- W3 is a "distributed heterogeneous collaborative multimedia information system"[1].

La definizione di Web fornita dal sito del W3C Consortium è:

- The World Wide Web (known as "WWW", "Web" or "W3") is the universe of network-accessible information, the embodiment of human knowledge[2] .

In effetti indipendentemente dalla definizione che si vuole scegliere il punto chiave è che il World Wide Web è stato in grado di fornire, tramite la rete di computer, un permanente accesso ad una varietà di informazioni, disponibili in diversi formati, tramite un aspetto molto semplice. Usando un interfaccia software sul web, come Internet Explorer, Mozilla, Netscape, Lynx, etc, il WWW ha cambiato il modo con cui la gente crea e visualizza le informazioni, è stata creata la prima vera rete ipertestuale[3] globale.

Alla base del successo del WWW ci sono state molte sue caratteristiche, tra le quali la possibilità di raggiungere le informazioni da tutto il mondo, informazioni che possono essere nei formati più diversi, come esempio testo, figure, immagini, suoni, filmati, etc. Inoltre il WWW ha fornito tramite dei collegamenti la possibilità di connettere informazioni, spesso residenti in posti diversi, in maniera molto semplice. La semplicità d'uso è sicuramente un fattore chiave che ne ha permesso la diffusione e lo sviluppo che oggi

[1] http://www.w3.org/Talks/General.html (tda) Il W3 (il Web) è un "sistema distribuito, etereogeneo, collaborativo di gestione delle informazioni multimediali"

[2] http://www.w3.org/WWW/ (tda) Il World Wide Web (conosciuto come "WWW", "Web" o "W3") è l'universo di informazioni accessibili via rete, l'incorporamento delle conoscenze umane.

[3] La definizione di ipertesto verrà data più avanti per ora ci base sapere che l'ipertesto è un testo con dei riferimenti altre informazioni

conosciamo, infatti è sufficiente avere a disposizione un computer collegato ad internet con un browser web per poter navigare. A questa caratteristica è inoltre associata la leggibilità universale del web, ovvero la possibilità di potersi connettere ad internet tramite qualunque tipo di computer indipendentemente dal tipo di computer con cui le informazioni sono state scritte e dal computer dove le informazioni risiedono. Gli elementi che hanno reso il Web molto popolare non sono però nati insieme al web ma derivano da idee precedenti, idee riutilizzate poi a favore delle sviluppo del web stesso, tratteremo queste origini nel prossimo paragrafo.

Approfondimenti

- What are CERN's greatest achievements? - The World Wide Web http://public.web.cern.ch/Public/Content/Chapters/AboutCERN/Achievements/WorldWideWeb/WWW-en.html
- Seminario online sul World Wide Web, 1991 Tim Berners-Lee http://www.w3.org/Talks/General.html

2.3 Le origini del web

Figura 1-Vannevar Bush

I concetti che stanno al base del Web non sono molto nuovi, infatti alcune idee che ispireranno il WWW nascono nel 1945. Nel 1945, Vannevar Bush, uno studioso americano, pubblica un articolo intitolato "As We May Think" sul giornale "Atlantic Monthly". Nell'articolo, Bush esponeva l'idea di una macchina capace di memorizzare informazioni testuali e grafiche in modo tale che ogni pezzo di informazione potesse essere collegato con gli altri. La macchina immaginaria era chiamata MEMEX. Pensandoci un attimo non è molto diverso da quello che il web ha fatto circa 45 anni dopo.

In particolare non essendo ancora disponibili apparecchiature elettroniche MEMEX proponeva un dispositivo basato su un meccanismo foto-elettrico-meccanico per la memorizzazione di documenti, oltre alla possibilità di poter creare e seguire collegamenti tra documenti memorizzati su microfilm.

MEMEX era quindi il primo esempio di information retrieval system (sistema di reperimento delle informazioni), nato ancora prima di poter utilizzare l'elettronica e quindi i computer come strumento di lavoro. Inoltre nell'articolo si proponeva anche di creare dei collegamenti tra documenti, in pratica quelli che oggi chiamiamo collegamenti ipertestuali. Queste sue idee saranno riprese ed ampliate da importanti studiosi negli anni seguenti ed a Vannevar Bush si ispirerà anche Tim Berners-Lee l'inventore del WWW.

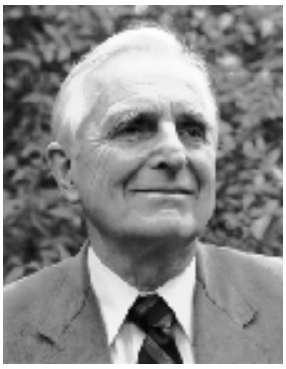


Figura 2-Douglas Engelbart

La storia delle idee che hanno portato al web passa anche da "oNLine System" (NLS) [NLS] un sistema proposto da Douglas Engelbart [ENGELBART] nel 1962. Engelbart fu uno dei seguaci dell'idea di Bush di creare una macchina che potesse aiutare l'uomo nel maneggiare le informazioni a sua disposizione. Engelbart lavorò per l'Ames Aeronautical Lab e per l'Augmentation Research Centre (ARC) a Standford dove creò con alcuni colleghi (William K. English e John F. Rulifson) On-Line System (NLS), la prima implementazione al mondo di quello che fu poi chiamato ipertesto. Nel 1962 fù pubblicato l'articolo Augmenting Human Intellect: A Conceptual Framework [AHUCF], nel quale veniva proposto "oNLine System (NLS)", un sistema che introduceva importanti innovazioni tra cui: - il mouse - la modifica di documenti su monitor - il collegamento tra

oggetti (link tra documenti) - il protocollo per terminali multipli - la teleconferenza - il word processing e le presentazioni

La demo del progetto NLS sarà pronta nel 1968 ed ancora oggi su internet si trovano i video [NSLDEMO] che mostrano i primi prototipi delle innovazioni sopra citate.

Figura 3-Ted Nelson

Nel 1965, quindi qualche anno dopo la presentazione di NLS, Ted Nelson [NELSON] conia il termine hypertext (ipertesto); l'ipertesto secondo la definizione di Nelson è un testo con collegamenti (link). Il termine ipertesto non è, come si potrebbe pensare, un termine prettamente informatico, infatti una enciclopedia con riferimenti ad altri termini è essa stessa un esempio di ipertesto. Gli studi di Nelson sugli ipertesti nascono da un progetto, nato nel 1960, di un sistema integrato di gestione della letteratura, ovvero dei contenuti, dei riferimenti espliciti ed impliciti, e del processo che sostiene la produzione letteraria (creazione, pubblicazione, modello economico). Il sistema (chiamato Xanadu [XANADU]), come evoluto alla metà degli anni Ottanta, era basato su server che costruivano documenti virtuali basati su pezzi di testo di lunghezza arbitraria, e client che permettevano la creazione di link.

Approfondimenti

- A Little History of the World Wide Web
  http://www.w3.org/History.html
- As We May Think - by Vannevar Bush
  http://www.theatlantic.com/doc/194507/bush

2.4 La nascita del WWW

Alla fine degli anni '80 Il CERN (European Organization for Nuclear Research) [CERN] [4]doveva affrontare il problema della perdita di informazioni dovuta principalmente al fatto che le informazioni erano distribuite su sistemi diversi consultabili con programmi e competenze specifiche. Ogni gruppo di lavoro aveva propri sistemi per la gestione delle informazioni ed i sistemi non potevano interagire tra loro.

Anche se gli scienziati del CERN spendevano molto tempo nella sede del CERN, spesso lavoravano in università e laboratori dislocati nelle loro nazioni. Avere un buon sistema di scambio di informazioni era chiaramente fondamentale. Insieme ai mezzi tradizionali (pubblicazioni su riviste internazionali, conferenze, scambio di idee nelle "coffee room", ...) gli scienziati del CERN avevano cominciato da anni a scambiarsi le informazioni in maniera elettronica, distribuendo in questo formato dati, software e documentazione. Inoltre dal 1980 in poi l'utilizzo della posta elettronica contribuì molto allo scambio di informazioni. Nonostante la possibilità di scambiare informazioni in formato elettronico negli anni '80 c'erano comunque molti ostacoli allo scambio effettivo dei dati. Erano presenti una vasta varietà di tipi di computer e di sistemi di rete con praticamente nessuna caratteristica in comune, inoltre le informazioni erano di tipo diverso e potevano essere accedute in modi molto diversi tra loro, creando una forte sensazione di sconforto ed inefficienza negli utenti. Tutto questo creò terreno fertile all'invenzione del World Wide Web, il cui scopo principale era permettere agli scienziati di accedere alle informazioni derivanti da qualunque tipo di sistema in una maniera coerente e semplice[5].

Il CERN necessitava quindi di un modello che rispondesse alle problematiche sopra riportate ed in generale che avesse i seguenti requisiti:

- essere accessibile via rete
- permettere l'accesso da sistemi eterogenei (VM/CMS, Macintosh, VAX/VMS, Unix)
- essere non centralizzato
- fornire l'accesso a dati esistenti
- avere aree private
- avere l'accesso a dati anche in fase di "lavori in corso"

---

[4] CERN Home page http://www.cern.ch

[5] What are CERN's greatest achievements? - Why from CERN? http://public.web.cern.ch/Public/Content/Chapters/AboutCERN/Achievements/WorldWideWeb/WhyCERN/WhyCERN-en.html

Tim Berners-Lee [Berners-Lee], un giovane fisico che lavorava al CERN, propose in un rapporto interno un sistema distribuito per la gestione delle informazioni che rispondeva ai requisiti sopra citati. Il modello proposto è illustrato nelle figura 5 nella quale si vede che l'informazione era posta al centro del modello ed indicata in con la dicitura "This document"; il documento era descritto tramite un ipertesto (la nuvoletta "Hypertext") e descriveva la proposta Mesh (la nuvoletta "Mesh") la quale a sua volta proponeva un sistema di unificazione dei CERNDOC, VAX/NOTES... Il documento ipertestuale (Hypertext) inoltre includeva altre informazioni riferite tramite collegamenti ipertestuali.

Figura 4 –Tim Bernes-Lee

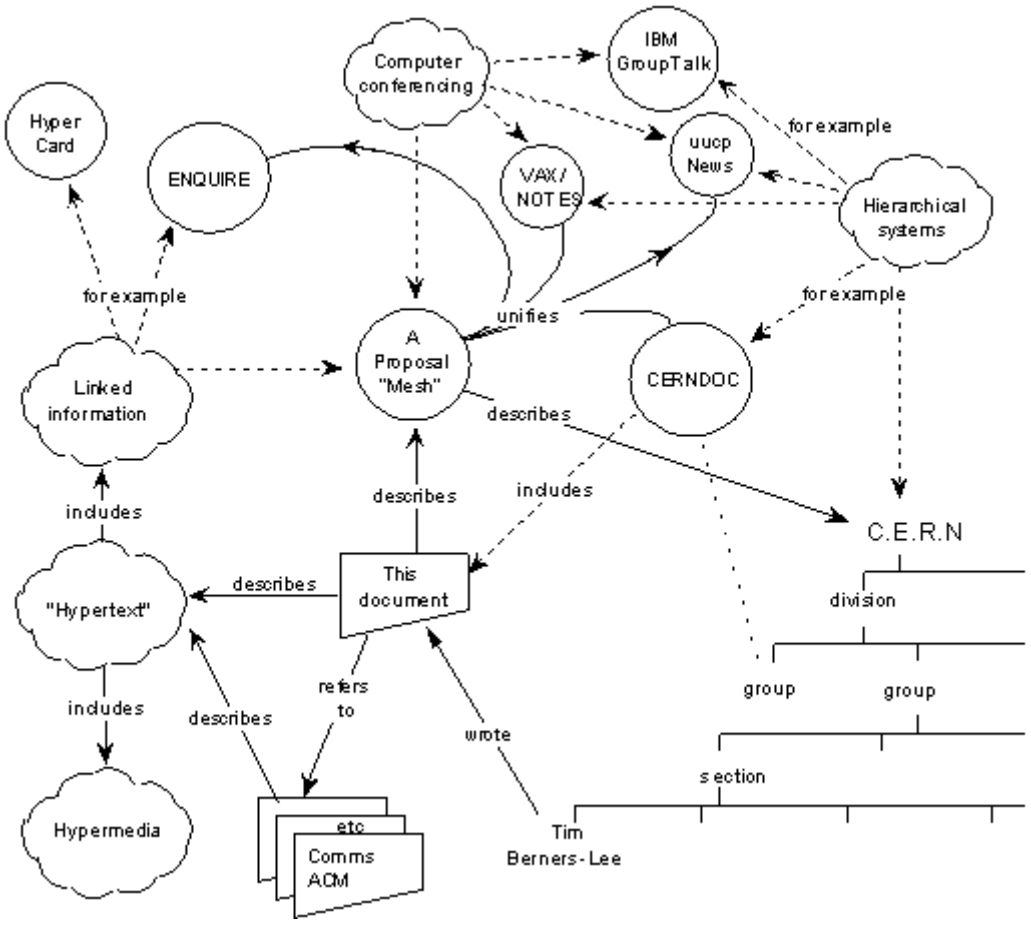


Figura 5 - Tim Berners-Lee propone un sistema distribuito di gestione delle informazioni

Tim Berners-Lee e Robert Cailliau per rafforzare la loro proposta scrivono il primo browser ipertestuale che fu chiamato WorldWideWeb. Il WorldWideWeb aveva integrato sia il browser che l'editor ed aveva quindi anche la possibilità di creare nuovi documenti e di modificare quelli esistenti tramite un'interfaccia visuale. Screenshot del WorldWideWeb sono mostrate sia in Figura 6 che in Figura 7. Si può notare anche come si potessero creare link tra i documenti e modificare lo stile delle pagine.

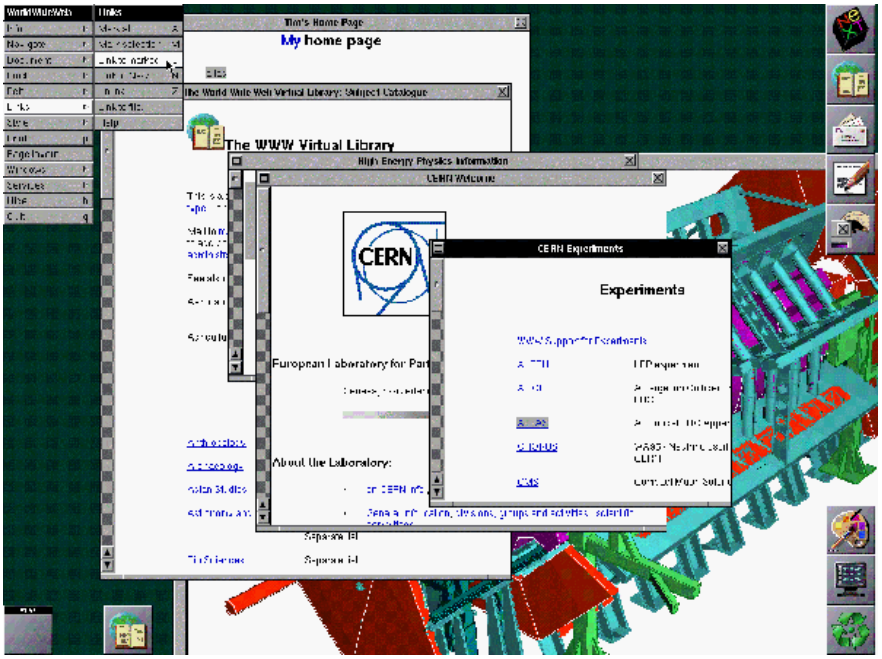


Figura 6 - Screenshot del browser WorldWideWeb

Figura 7 - WorldWideWeb: Browser ed Editor

Approfondimenti

- The original proposal of the WWW, HTMLized http://www.w3.org/History/1989/proposal.html

2.5 Storia del WWW

Gli anni 1989 e 1990 sono stati un periodo di divulgazione del progetto del WWW, progetto che finalmente dal 1991 in poi si è evoluto ad una velocità impressionante. Essendo molto difficile descrivere la storia del web dal 1991 in poi di seguito sono riportati gli avvenimenti principali che si sono susseguiti di anno in anno permettendo lo sviluppo del web fino a quello che oggi conosciamo.

1991 nascono i primi browser e server multi-piattaforma; viene organizzato il primo seminario sul WWW;

1992 il browser "Viola" [VIOLA] aggiunge la possibilità di vedere immagini (oltre il testo);

1993 inizio dell'anno: ci sono oltre 50 server web conosciuti; settembre: NCSA [NCSA] rilascia il browser Mosaic; fine dell'anno: incremento dell'utilizzo del WWW, ci sono oltre 200 server web conosciuti;

1994 aprile: fondata la Mosaic Communication Corporation che produce il browser Mosaic Netscape (il primo Netscape); maggio: si tiene la prima conferenza internazionale sul WWW; fine dell'anno: Rilasciata la versione 2.0 di Netscape; (integrato il client di posta ed il newsreader, supporto per i frames, i div, compatibilità con le gif, etc.);

1995 aprile: viene rilasciata la prima versione pubblica (la 0.6.2) di Apache [APACHE]; rilasciato Internet Explorer 2;

1996 Microsoft produce Internet Explorer 3, Netscape Communication produce Netscape 3 (comincia il supporto ai CSS);

1997 nasce XML, il linguaggio di markup che diventerà uno dei mattoni fondamentali del web; rilasciati Netscape 4, Explorer 4, Opera 3 (cresce il supporto ai CSS, nasce il DHTML ed il DOM, Microsoft introduce le ActiveX);

1998 quinta generazione dei browser (Explorer 5, Netscape 5). Nasce il progetto Mozilla [MOZILLA].

Dal 1998 in poi la ricerca sul Web si sposterà verso la famiglia delle tecnologie XML. Le tecnologie HTML e CSS avranno una minore evoluzione ma in compenso si svilupperanno quelle tecnologie che renderanno possibile una maggiore integrazione delle varie tecnologie web; tra le novità che verranno introdotte possiamo per esempio citare i web services, il semantic web, XForm, XQuery, XSL/XSLT, l'iniziativa WAI, etc. Il lettore interessato alle tecnologie basate su XML può sicuramente trovare molta documentazione sul sito del W3C [W3C].

Approfondimenti

- W3C 10th anniversary - timeline graphic http://www.w3.org/2005/01/timelines/timeline-2500x998.png
- quirksmode.org - A history of browsers http://www.quirksmode.org/browsers/history.html
- Extensible Markup Language (XML) http://www.w3.org/XML/

2.6 Il web ed internet

Il web ed internet[6] sono due cose distinte, il web è solamente uno dei servizi di comunicazione basati su internet. La relazione tra le due entità può essere spiegata con una semplice analogia utilizzando la rete stradale. In internet, come sulla rete stradale, sono essenziali tre elementi: la connessione fisica (i cavi e le strade), i comportamenti comuni (il codice stradale e i protocolli di rete) e i servizi (il servizio di posta terrestre e il WWW).

2.6.1 Le connessioni fisiche: cavi e strade

I cavi sono infrastrutture passive, mantenute da enti pubblici e da compagnie di telecomunicazioni. I cavi hanno differenti capacità: una singola linea telefonica come le normali linee telefoniche di casa possono trasportare circa 56 kilobytes al secondo, l'equivalente di una pagina di testo con qualche immagine per secondo. Le fibre ottiche possono trasportare migliaia di milioni di bytes al secondo. Anche se i cavi possono essere di differenti tipi e le loro congiunzioni possono essere molto complicate, questi sono tutti interconnessi tra di loro. Per tornare alla nostra analogia, sulle nostre strade è possibile spostarsi da una città all'altra passando attraverso tipi di strade diverse: strade urbane, extraurbane, autostrade, etc, e tutte queste strade sono interconnesse tra di loro. Tramite le diverse connessioni fisiche è quindi per esempio possibile raggiungere con il nostro computer di casa un computer in Australia o in America, attraverso una serie contigua di collegamenti fisici e di nodi di interscambio che permettono di passare da un collegamento all'altro.

2.6.2 Le regole di comportamento comuni

Connettere un computer al sistema di connessioni di internet non è però sufficiente: per poter dialogare con gli altri è necessario concordare con loro delle regole di comportamento (di comunicazione), così come noi facciamo quando guidiamo sulle strade. Internet è come il codice delle strada: i computer devono usare i collegamenti fisici in accordo a delle regole concordate. Migliaia di macchine possono utilizzare le strade anche se hanno destinazioni diverse, non ci saranno problemi fino a quando ognuno

[6] http://public.web.cern.ch/Public/Content/Chapters/AboutCERN/Achievements/WorldWideWeb/WebInternet/WebInternet-en.html

rispetterà le regole del codice stradale. Internet trasferisce le informazioni in piccoli pacchetti scambiati tra computer. Per poter scambiare questi pacchetti i computer sono soggetti a delle regole: devono usare lo stesso protocollo di comunicazione. Un protocollo di comunicazione è qualcosa con cui tutti siamo abituati ad aver a che fare quando abbiamo una conversazione con qualcuno: durante la conversazione le persone sanno quando possono cominciare a parlare, quando fermarsi, come fare per incoraggiare le altre persone a cominciare a parlare e così via; questo è il protocollo di comunicazione "implicito" utilizzato dagli umani. I computer scambiano le informazioni utilizzando un insieme di regole e di comportamento, per essere connesso ad internet un computer deve rispettare i protocolli di internet. Il protocollo più comune utilizzato è il Transmission Control Protocol / Internet Protocol o TCP/IP[7].

2.6.3 Servizi per tutti

Una volta che si hanno le connessioni fisiche ed i protocolli di rete è possibile comunicare con gli altri computer. Ma cosa possono comunicarsi i computer tra loro? Le strade possono essere utilizzate per guidare da solo, per viaggiare sui bus con le altre persone, per trasportare materie prime, la posta terrestre, etc. Allo stesso modo tramite internet possono funzionare diversi servizi: la posta elettronica, il trasferimento di file, le connessioni a server remoti, etc. il World Wide Web è appunto uno dei servizi che funziona tramite internet, può essere visto come una distribuzione personalizzata delle informazioni: alla tua richiesta, il WWW ti consegna il documento richiesto.

Approfondimenti

- Internet Society - Home page http://www.isoc.org/
- The Internet Engineering Task Force - Home page http://www.ietf.org/
- Internet Assigned Numbers Authority (IANA) - Home page http://www.iana.org/
- Internet Society - All about the internet http://www.isoc.org/internet/infrastructure

[7] A TCP/IP Tutorial, gennaio 1991 http://www.ietf.org/rfc/rfc1180.txt

2.7 Come funziona il web

Il web è l'universo di informazioni disponibili con un click del mouse[8]. Per usarlo c'è bisogno di un computer, una connessione a internet ed un browser.

Quando utilizzi il tuo browser, lui trova e mostra pagine di informazioni. Il ruolo del browser è quello di interpretare il linguaggio delle pagine web (HTML, ...) e trasformarlo in testo e immagini che puoi vedere sul tuo schermo. Se hai bisogno di ulteriori informazioni, tutto quello che hai da fare è cliccare su un iperlink.

I documenti sono memorizzati in computer chiamati "server" i quali forniscono i documenti agli utenti che possono visualizzarli richiedendoli tramite il loro computer, chiamato client. Questo modello di computer che effettuano delle richieste e computer che rimangono in attesa di richieste e forniscono le riposte è chiamato modello client/server. I computer che partecipano a questo meccanismo sono naturalmente collegati tramite internet. Quindi quando tramite il tuo browser clicchi su un iperlink, il tuo computer richiede ad un server di restituirgli un documento che ti verrà poi mostrato nel browser.

Per esempio partendo dalla homepage del CERN http://www.cern.org in svizzera, puoi cliccare ed ottenere un documento che è in un laboratori di fisica dall'altra parte del mondo. Tutte le informazioni sembrano essere in una piccola scatola davanti a te ma in realtà sono sparse in tutto il mondo.

Il web fà anche un buon uso della rete (internet): quando clicchi su iperlink il tuo browser "ordina un documento" ad un altro computer, lo riceve e poi te lo mostra. Tu sei poi libero di guardarlo subito oppure quando vuoi, senza ulteriore consumo di risorse di rete.

Il web può essere utilizzato per iniziare dei processi sia lato client che lato server. Una richiesta può per esempio avviare un ricerca in un database di un server, il quale restituisce un documento generato tramite i i risultati della ricerca. Un documento restituto ad un client può essere in un formato che induce il browser ad avviare un programma esterno al browser stesso per interpretarlo (come per esempio Acrobat Reader per il file PDF). L'abilità del web di negoziare il formato delle informazioni tra client e server rende possibile poter scambiare informazioni di qualunque tipo se il client possiede il software appropriato per poter maneggiare l'informazione ricevuta. Questo rende possibile che video, suono ed ogni altro formato sia accessibile agli utenti senza la necessità che una singola applicazione (il browser) debba essere in grado di interpretare ogni cosa.

[8] How the Web works http://public.web.cern.ch/Public/Content/Chapters/AboutCERN/Achievements/WorldWideWeb/HowWorks/HowWorks-en.html

Riassumendo gli attori del web sono:

- gli utenti (user) le persone che accedono alle informazioni
- i browser i programmi utilizzati per accedere alle informazioni sul web.
- i client i dispositivi con i quali si accede alle risorse del WWW (computer, palmari, telefoni, etc). Con client, a seconda del contesto di può intendere il browser, oppure il dispositivo.
- server i computer attraverso il quali si mettono a disposizione delle risorse (informazioni o servizi) ad altri server o ai vari client

2.8 Principi fondamentali del WWW

Quello che Tim Berners-Lee fece quando ideò il web fu quello di utilizzare alcuni concetti basi dai sui predecessori (V. Bush, D. Engelbart, T. Nelson, etc) e di renderli possibili tramite protocolli e tecnologie semplici da utilizzare.

I concetti fondamentali sui quali il web si basa sono:

- La leggibilità universale
- Gli ipertesti
- La ricercabilità dei documenti
- Il modello client-server
- La negoziazione del formato

2.8.1 Leggibilità universale

È stata uno dei principi ispiratori del WWW che ha permesso l'integrazione di informazioni derivanti da fonti diverse. Le informazioni disponibili sul web possono essere reperite:

- da ogni tipo di computer
- da ogni paese nel mondo tramite il web
- da qualunque persona (autorizzata)
- tramite un semplice programma di navigazione

2.8.2 Gli ipertesti

Gli ipertesti secondo la definizione data da Nelson sono dei organizzati in modo non lineare, cioè che danno, tramite gli iperlink, la possibilità di saltare da una parte all'altra del testo senza seguire un percorso lineare. Il web in realtà è rimasto per molto poco tempo solo con testo ma il concetto di ipertesto si è evoluto in quello di ipermedia, quindi non solo testo ma anche immagini, video, etc. Gli ipertesti, e gli ipermedia permettono di:

- navigare tra le informazioni in modo non sequenziale.
- riferire informazioni dislocate in posizioni diverse

2.8.3 La ricercabilità delle informazioni

I documenti messi a disposizione tramite il Web dovevano essere ricercabili. Il web nasce già con le basi per lo sviluppo dei motori di ricerca e questo è permesso da:

- reperibilità permanente dei documenti
- accessibilità dei documenti da parte di indicizzatori di contenuti.

2.8.4 Modello client-server

Come abbiamo visto nel paragrafo precedente il web segue il modello client-server dove le informazioni sono:

- memorizzate sui Server Web
- raggiungibili tramite un client Web

I client effettuano richieste ai server Web per ottenere le informazioni desiderate.

2.8.5 La negoziazione del formato

Nel modello del Web si doveva permettere lo scambio di documenti memorizzati nei formati più diversi. Il meccanismo della negoziazione del formato permette di scambiare tramite il web file di vario tipo, tra cui: HTML, PDF, immagini (png, jpg, gif), audio, video. La negoziazione del formato è il meccanismo tramite il quale insieme alle informazioni che si scambiano sulla rete (file di vario tipo, png, jpg, etc.) viene anche inviato il tipo dell'informazione che stiamo passando (il content-type) attraverso il quale i vari client possono decidere l'applicazione che devono utilizzare per "trattare" il contenuto ricevuto. Per esempio se il client riceve un file con content-type "application/pdf" probabilmente tenterà di visualizzare il file con un visualizzatore di PDF.

2.8.6 Le tecnologie ed i protocolli fondamentali del WWW

Il WWW si basa su alcuni protocolli e tecnologie fondamentali:

- HTTP (HyperText Trasfer Protocol)
- URI, URL (Uniforme Resource Identifier, Uniforme Resource Locator)

- HTML (HyperText Markup Language)

Queste tre tecnologie stanno alla base del funzionamento del Web e servono per scambiare le informazioni (HTTP), identificare univocamente le informazioni (URI, URL), e rappresentare le informazioni stesse (HTML). L'HTTP è un protocollo e come tale definisce il formato e l'ordine dei messaggi scambiati tra due o più entità, così come le azioni intraprese nella trasmissione o ricezione di un messaggio. Le URI (e URL) identificano le informazioni presenti sul web, mentre l'HTML definisce la rappresentazione dell'informazione. Nei paragrafi che seguono le tre tecnologie/protocolli verranno trattate in maggiore dettaglio.

Approfondimenti

- Wikipedia: Client-server http://en.wikipedia.org/wiki/Client_server

2.9 HTTP

Il web utilizza un proprio protocollo per trasferire le informazioni, l'HTTP (Hypertext Transfer Protocol). HTTP trasmette le proprie informazioni tramite una connessione TCP/IP stabilita tra due computer collegati tra di loro (tipicamente tramite internet, ma potrebbe anche essere una rete diversa come per esempio una rete di computer privata).

HTTP fornisce:

- un sistema veloce di trasferimento dati
- un sistema senza stati (stateless)
- un architettura estendibile
- un sistema per la negoziazione dei differenti tipi di dato trasferiti

Di seguito sono riportare le varie fasi della comunicazione client/server necessarie per comporre una pagina web. Nella prima fase (Figura 8) il client richiede al server il testo del documento (il suo codice HTML) all'interno del quale possono esserci riferimenti ad altri oggetti da richiedere al server (immagini, CSS, file esterni,...). Nella seconda fase (Figura 9) il client ogni qual volta trova nel testo del documento degli oggetti esterni necessari per visualizzare la pagina (come per esempio immagini) le richiede al server con un altra richiesta HTTP.

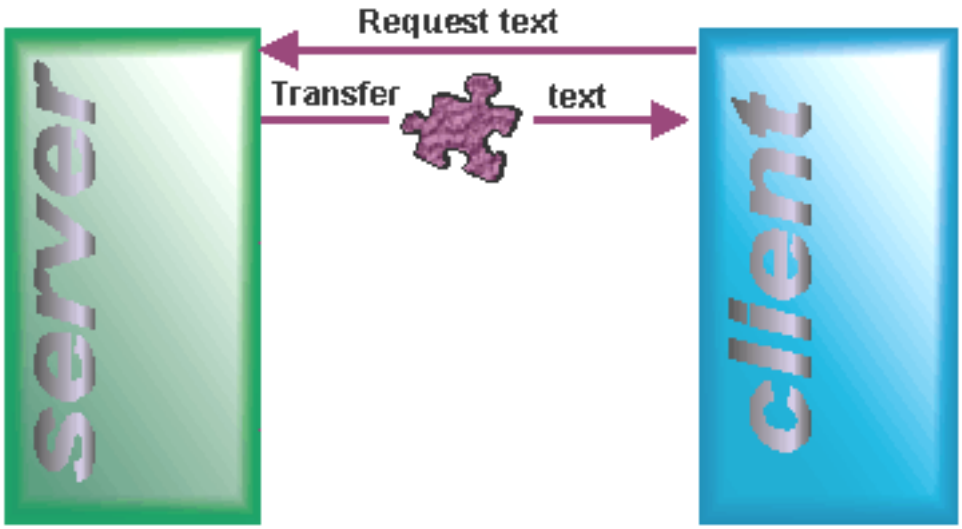


Figura 8 - Richiesta della pagina

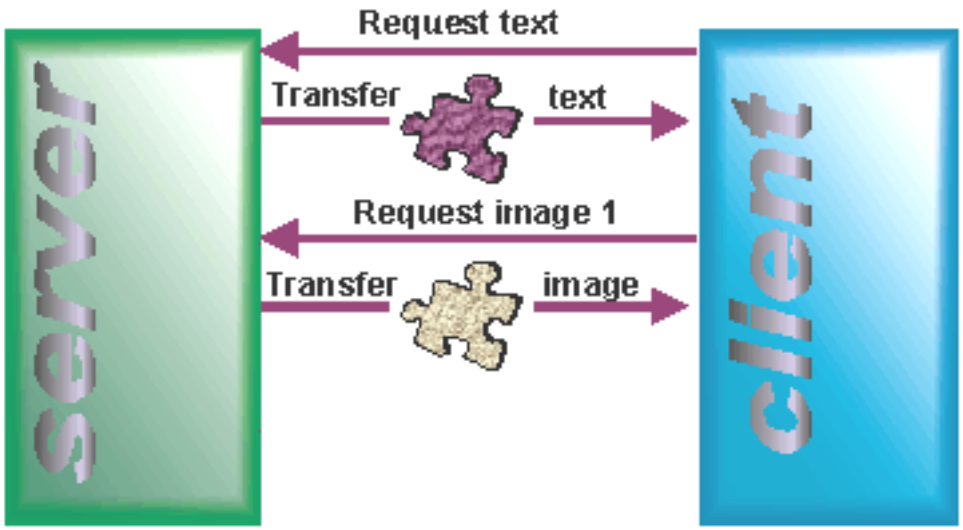


Figura 9 - Invio della pagina e richiesta degli altri elementi della pagina

Nella Figura 10 viene evidenziato che possono essere necessari molti oggetti per formare una pagina web completa. Infine (Figura 11) il client mostra all'utente la pagina composta da tutti gli elementi richiesti.

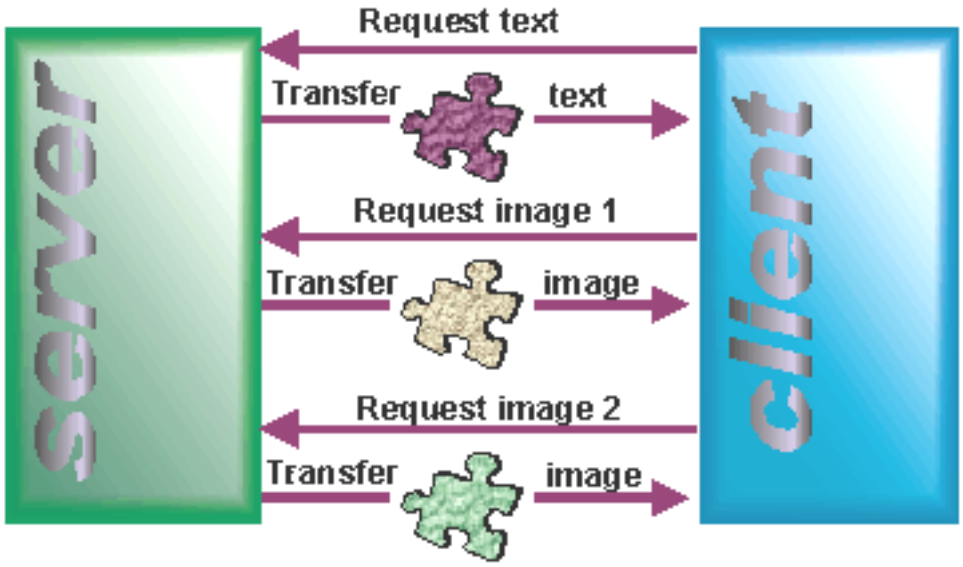


Figura 10 - Richiesta ed invio di tutti gli elementi della pagina

La Figura 11 mostra come, alla fine del processo di trasferimento dei vari file, il client riesce a ricostruire, e visualizzare, la pagina web completa.

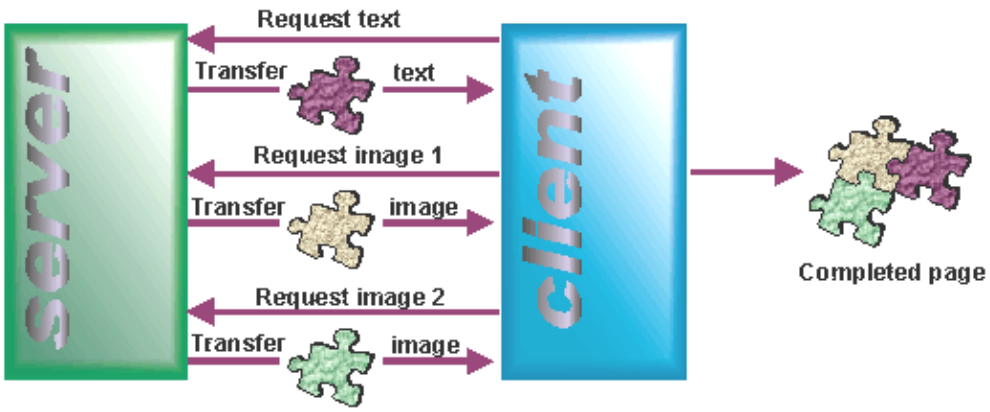


Figura 11 - Visualizzazione della pagina nel client

Approfondimenti

- Hypertext Transfer Protocol - HTTP/1.1, giugno 1999 http://www.ietf.org/rfc/rfc2616.txt

2.10 Breve storia dell'HTTP

L'HTTP nella sua versione originaria, del 1990, è conosciuto come HTTP 0.9. La versione HTTP 1.0 pubblicata nel 1995, formalizza l'uso comune del protocollo che si era sviluppato fino a quegli anni. L'attuale versione, HTTP 1.1, formalizzata nel 1996 e riedita nel 1999, è descritta nella RFC 2616 [RFC2616].

2.10.1 Cosa è una RFC?

Le RFC (Request for Comments) sono una serie di documenti tecnici e note organizzative che trattano di Internet (originariamente ARPANET[9]), cominciate nel 1969. Le RFC discutono una serie di aspetti delle reti di computer, tra cui protocolli, procedure, programmi, concetti, così come note ed opinioni. I documenti con le specifiche ufficiali dell'insieme dei protocolli Internet sono definite dall'Internet Enginnering Task Force e dal Internet Engineering Steerin Group (IESG) e sono registrate e pubblicate con il processo standard delle RFC [RFC]. Il risultato è che il processo di pubblicazione delle RFC gioca un ruolo importante nel processo di standardizzazione di Internet.

2.10.1.1 L'RFC-Editor

L'RFC Editor è il responsabile della pubblicazione delle RFC e della revisione editoriale finale dei documenti. RFC Editor mantiene anche un file chiamato "RFC Index" che può essere trovato online. Per oltre 30 anni, l'RFC Editor è stato Jon Postel[POSTEL], oggi l'RFC Editor è un piccolo gruppo fondato dalla "Internet Society (ISOC) [ISOC]".

[9] Arpanet è il precursore di Internet, una vasta rete di computer create da United States Defense Advanced Research Project Agency (ARPA). Arpanet nasce nel 1969 come sistema per lo studio di nuove tecnologie di interconnessione di rete tra università e centri di ricerca americani. Maggiori informazioni possono essere trovate sul sito http://www.darpa.mil.

2.11 HTTP 1.1

HTTP 1.1 è retro compatibile con HTTP/1.0; un Web server che utilizza HTTP 1.1 può parlare con un browser che utilizza la versione 1.0 e viceversa. Il protocollo HTTP (sia nella versione 1.0 che 1.1) fornisce le regole per le interazioni tra client e server, tra cui:

- Message i messaggi tra client e server (gli header, il corpo del documento)
- Request le regole per fare le richieste al server (Get, Post, Head, etc., i parametri accettati)
- Response le regole per fornire le risposte al client(le date di modifica, la cache, dimensioni, etc.)
- Connection i metodi per stabilire le connessioni tra client e server
- Status Code i codici di stato (200, 404, etc..)

2.11.1 HTTP 1.1: Status Code più comuni

I codici di stato sono molto utili per verificare il corretto dialogo tra client e server e quindi anche la validità delle nostre pagine HTML. I codici di stato (Status Code) più utilizzati sono:

200 - OK

La richiesta è corretta, l'output è restituito nel corpo del messaggio (message body).

404 - Not Found

La risorsa richiesta non esiste.

301 - Moved Permanently

La risorsa non è più disponibile a questo indirizzo.

500 - Server Error

Si è verificato un errore sul server. La causa più comune è uno script sul server che fallisce.

2.11.2 HTTP 1.1: categorie di Status Code

Gli Status code[10] [1] si suddividono in categorie:

1xx
: sono messaggi solo di tipo informativo

2xx
: indicano una richiesta in qualche modo soddisfatta

3xx
: re-direzionano il client ad un altro indirizzo

4xx
: indicano un errore da parte del client

5xx
: indicano un errore da parte del server

Elenco degli status code in HTTP 1.1

100	Continue	404	Not Found
101	Switching Protocols	405	Method Not Allowed
200	OK	406	None Acceptable
201	Created	407	Proxy Authentication Required
202	Accepted	408	Request Timeout
203	Non-Authoritative Information	409	Conflict
204	No Content	410	Gone
205	Reset Content	411	Length Required
206	Partial Content	412	Precondition Failed
300	Multiple Choices	413	Request Entity Too Large
301	Moved Permanently	414	Request URI Too Long
302	Moved Temporarily	415	Unsupported Media Type
303	See Other	416	Requested Range Not Satisfiable
304	Not Modified	417	Expectation Failed
305	Use Proxy	500	Internal Server Error
306	(not Used)	501	Not Implemented

[10] Trattazione completa degli status code in http può essere reperita all'indirizzo http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html.

100	Continue	404	Not Found
307	Temporay Redirect	502	Bad Gateway
400	Bad Request	503	Service Unavailable
401	Unauthorized	504	Gateway Timeout
402	Payment Required	505	HTTP Version Not Supported
403	Forbidden		

2.11.3 Richieste multiple

Una delle differenza principali tra l'HTTP 1.1 e le precedenti versioni è la possibilità di fare richieste multiple. Effettuando più richieste contemporaneamente si diminuisce il "round-trip time" globale ovvero il tempo che intercorre tra l'inizio della richiesta e la fine della risposta, come si vede dalla Figura 12 il tempo totale per servire tutte le richieste è molto minore in caso di richieste multiple.

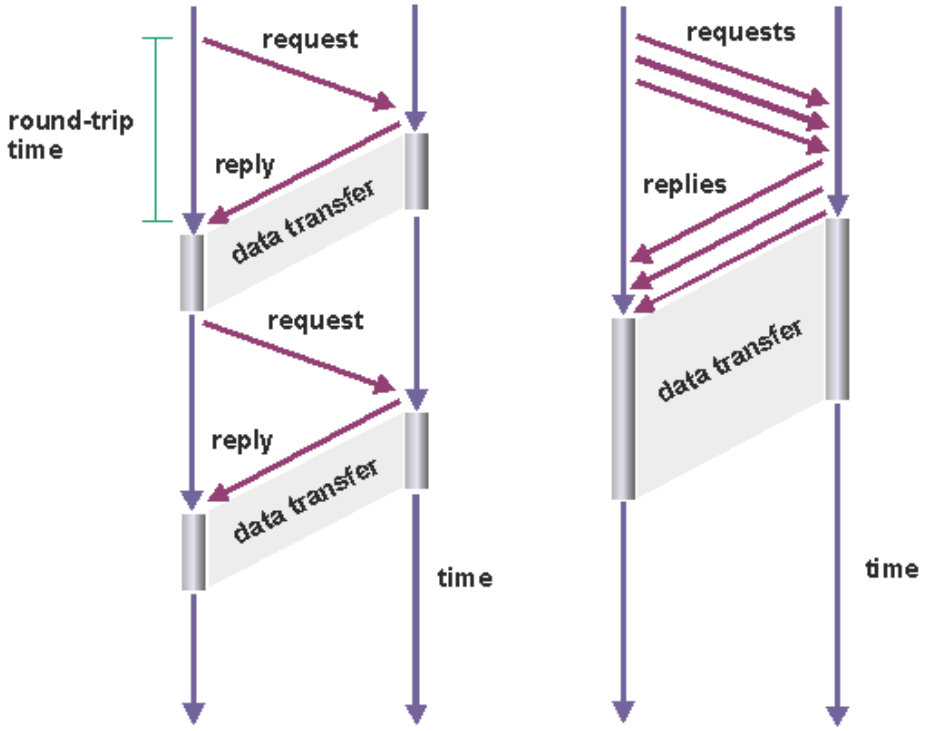


Figura 12 – Differenze nelle richeiste multiple tra HTTP1.0 (sinistra) e HTTP1.1 (destra)

2.11.4 Connessione permanente

Un'altra delle innovazioni dell'HTTP 1.1 rispetto alle precedenti versioni è la possibilità di mantenere una connessione permanente. Questa diminuisce notevolmente il tempo dovuto alle connessioni mantenendo la connessione aperta per farvi transitare tutte le altre connessioni virtuali, vediFigura 13.

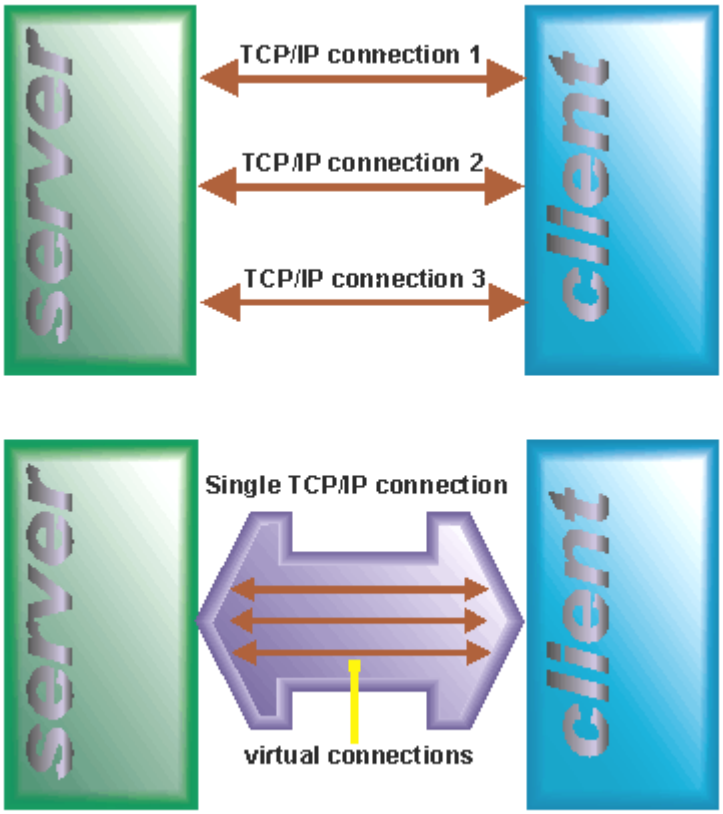


Figura 13 – Differenze tra connessioni singole e connessione permanente. HTTP1.0 (figura in alto) e HTTP1.1 (figura in basso)

2.12 URI

Le risorse sul web sono identificate tramite una URI[RFC3986] (Uniform Resource Identifier). L'URI, come dice il nome stesso, identifica in modo univoco una risorsa sul Web. Le URI possono essere utilizzate per identificare risorse diverse da pagine web, l'esempio di Figura 14 mostra come la risorsa "Oaxaca Weather Info" sia identificata tramite l'URI http://weather.example.com/oaxaca, mentre la sua rappresentazione sia la pagina HTML mostrata in figura. Le URI sono un concetto non legato solamente alle risorse Web, per esempio il codice ISBN di un libro è un esempio di un identificatore univoco (un URI) che però non si riferisce ad una risorsa sul web ma ad un libro.

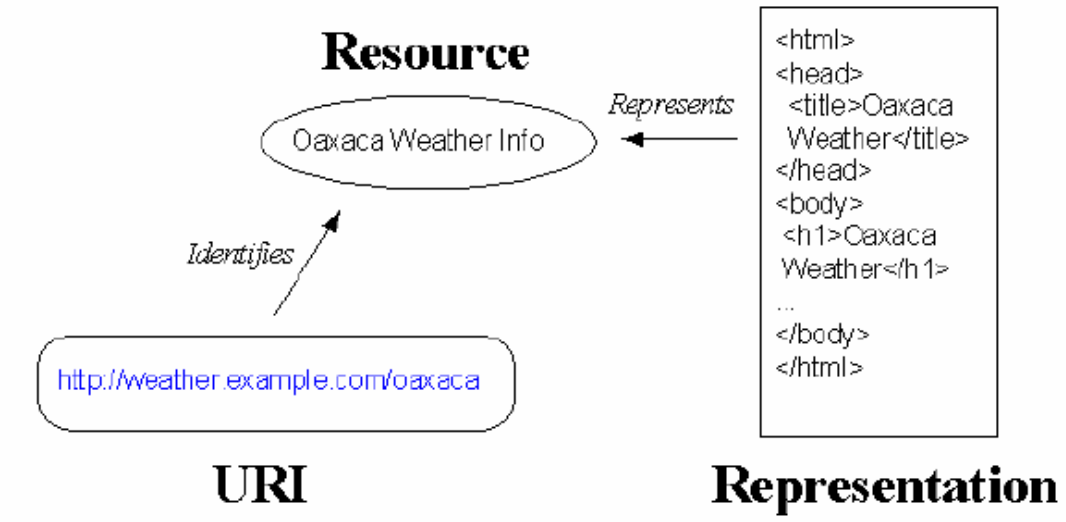


Figura 14 - Ruolo dell'URI nel WEB

2.12.1 Composizione delle URI

Le URI sono composte da:

- schema
- parte specifica dello schema

secondo la sintassi

```
<scheme>:<scheme-specific-part>
```

Alcuni esempi di URI sono:

- schema ftp per File Transfer Protocol services ftp://ftp.is.co.za/rfc/rfc1808.txt

- schema http per Hypertext Transfer Protocol services http://www.math.uio.no/faq/compression-faq/part1.html
- schema mailto per indirizzi di posta elettronica mailto:mduerst@ifi.unizh.ch

2.12.2 URI Schema

Le URI Schema definiscono la metodologia per individuare la risorsa in modo univoco. La <scheme-specific-part> viene interpretata in funzione di quale schema stiamo utilizzando. Il registro ufficiale dei nomi degli schema delle URI è mantenuto da IANA[IANA]. Esempi di schema registrati sono:

- ftp
- http
- file

Schemi di URI [URISCHEMES]

Nome schema	Descrizione	Riferimento
ftp	File Transfer Protocol	[RFC1738]
http	Hypertext Transfer Protocol	[RFC2616]
gopher	The Gopher Protocol	[RFC1738]
mailto	Electronic mail address	[RFC2368]
news	USENET news	[RFC1738]
nntp	USENET news using NNTP access	[RFC1738]
telnet	Reference to interactive sessions	[RFC1738]
wais	Wide Area Information Servers	[RFC1738]
file	Host-specific file names	[RFC1738]
prospero	prospero Directory Service	[RFC1738]
z39.50s	Z39.50 Session	[RFC2056]
z39.50r	Z39.50 Retrieval	[RFC2056]
cid	content identifier	[RFC2392]
mid	message identifier	[RFC2392]
vemmi	versatile multimedia interface	[RFC2122]
service	service location	[RFC2609]
imap	internet message access protocol	[RFC2192]
nfs	network file system protocol	[RFC2224]
acap	application configuration access protocol	[RFC2244]
rtsp	real time streaming protocol	[RFC2326]
tip	Transaction Internet Protocol	[RFC2371]
pop	Post Office Protocol v3	[RFC2384]
data	data	[RFC2397]
dav	dav	[RFC2518]
opaquelocktoken	opaquelocktoken	[RFC2518]
sip	session initiation protocol	[RFC2543]
tel	telephone	[RFC2806]
fax	fax	[RFC2806]
modem	modem	[RFC2806]

ldap	Lightweight Directory Access Protocol	[RFC2255]
https	Hypertext Transfer Protocol Secure	[RFC2818]
soap	soap.beep	[RFC3288]
soap.beeps	soap.beeps	[RFC3288]
xmlrpc.beep	xmlrpc.beep	[RFC3529]
xmlrpc.beeps	xmlrpc.beeps	[RFC3529]
urn	Uniform Resource Names	[RFC2141]
go	go	[RFC3368]
h323	H.323	[RFC3508]
ipp	Internet Printing Protocol	[RFC3510]
tftp	Trivial File Transfer Protocol	[RFC-lear-tftp-uri-06.txt]

2.12.3 URI HTTP

Le URI http sono così composte[11]: http_URI = "http:" "//" [authority "@"] host [":" port] [abs_path]

- authority informazioni utili per l'autorizzazione
- host un nome a dominio o un indirizzo IP
- port un numero intero
- abs_path identificativo per raggiungere la risorsa sull'host indicato

Esempi

http://www.ietf.org/rfc/rfc2396.txt
http://lucchesi:password@www.iit.cnr.it:80/index.php

2.12.4 Struttura ad albero degli URI http

Le URI http hanno una struttura ad albero gerarchica delimitata da "/".

Esempio di struttura gerarchica

`http://www.w3.org/TR/xhtml1/Overview.html`

individua la risorsa tramite lo schema http scomponendo l'identificativo in:

- www.w3.org
- TR
- xtml1
- Overview.html

[11] Ricordarsi che le parti fra parentesi quadre ([...]) non sono obbligatorie.

2.12.5 Elementi riservati, e valori predefiniti nelle URI

Alcuni caratteri nelle URI sono riservati poiché coprono particolari esigenze.

I caratteri che seguono sono alcuni tra quelli riservati:

";" | "/" | "?" | ":" | "@" | "&" | "=" | "+" | "$" | ","

Esempio:

"/" è utilizzato come separato negli schema delle URI con struttura gerarchica.

Le URI HTTP hanno dei valori predefiniti:

- se la porta non è esplicitamente indicata allora la porta viene impostata a 80
- se abs_path non è esplicitamente indicato allora l'abs_path viene impostato a /

2.12.6 URL

La URL è un concetto informale per identificare un tipo di URI.

Le URL sono URI che identificano una risorsa tramite la rappresentazione del metodo di accesso alla risorsa stessa (normalmente l'indirizzo di rete della risorsa).

Esempi di URI:

http://www.ietf.org/rfc/rfc2396.txt	è una URL (quindi anche una URI !)
ISBN:0-201-59625-3	è una URI, ma **non** è una URL

Nota: Contemporary View

Over time, the importance of this additional level of hierarchy seemed to lessen; the view became that an individual scheme did not need to be cast into one of a discrete set of URI types, such as "URL", "URN", "URC", etc. Web-identifier schemes are, in general, URI schemes, as a given URI scheme may define subspaces. Thus "http:" is a URI scheme. "urn:" is also a URI scheme; it defines subspaces, called "namespaces". For example, the set of URNs, of the form "urn:isbn:n-nn-nnnnnn-n", is a URN namespace. ("isbn" is an URN namespace identifier. It is not a "URN scheme", nor is it a "URI scheme.")

Further, according to the contemporary view, the term "URL" does not refer to a formal partition of URI space; rather, URL is a useful but informal concept. A URL is a type of URI that identifies a resource via a representation of its primary access mechanism (e.g., its network "location"), rather than by some other attributes it may have. Thus, as we noted, "http:" is a URI scheme. An http URI is

a URL. The phrase "URL scheme" is now used infrequently, usually to refer to some subclass of URI schemes which exclude URNs.

Approfondimenti

- Naming and Addressing: URIs, URLs, ...
  http://www.w3.org/Addressing/
- Uniform Resource Identifiers (URI): Generic Syntax, agosto 1998
  http://www.ietf.org/rfc/rfc2396.txt
- Internationalized Resource Identifiers (IRIs), gennaio 2005
  http://www.ietf.org/rfc/rfc3987.txt

2.13 Linguaggi di markup

I linguaggi di markup sono stati scelti per rappresentare dei documenti, identificando delle porzioni di testo tramite dei marcatori (i tag). L'HTML, insieme a XML [XML], è forse il più famoso dei linguaggi di markup, esso trae le sue origini da un altro linguaggio di markup: SGML [SGML] (Standard Generalized Markup Language). SGML fu definito come standard nel 1986 e già portava con se i principi che avrebbero ispirato l'HTML: struttura gerarchica del documento separazione tra contenuto e presentazione intellegibile dalle persone ed interpretabile dalle macchine.

Esempio di codice di markup

```
<libro>
   <titolo>Divina Commedia</titolo>
   <autore>Dante Alighieri</autore>
   <capitoli>
     <capitolo>
       Nel mezzo del cammin di .....
     </capitolo>
   </capitoli>
</libro>
```

La presentazione dei contenuti nei linguaggi di markup è di solito lasciata ad un linguaggio esterno, in SGML, per esempio, si usa DSSSL [DSSSL] (Document Style Semantics and Specification Language), in XML si usa XSL [XSL] (XML Stylesheet language), in HTML su usa CSS (Cascading Style Sheet).

2.14 HTML

HTML è un linguaggio di markup nato dall'SGML nel 1990 e ideato Tim Berners-Lee. SGML tramite una DTD (Document Type Definition) regolamenta la struttura del documento: gli elementi permessi gli attributi permessi la possibile struttura di un documento L'HTML nasce come una classe SGML con una propria DTD. Nella prima versione del HTML (conosciuta con HTML 1.0) erano presenti pochissimi elementi, mentre molti di quelli attuali sono stati aggiunti con il tempo. L'HTML permette di separare la struttura del documento dalla presentazione e di effettuare gli iperlink (i collegamenti tra ipertesti o ipermedia). Il fatto che HTML sia SGML valido apre la porta allo scambio di informazioni tra sistemi diversi. HTML infatti descrive la struttura logica di un documento, non la sua formattazione, la quale può essere adattata in funzione delle diverse piattaforme utilizzando diversi font e convenzioni.

Approfondimenti

- HyperText Markup Language (HTML) Home Page
  http://www.w3.org/MarkUp/
- w3schools: Introduction to DTD
  http://www.w3schools.com/dtd/dtd_intro.asp

2.15 Non c'è solo HTML

Nel tempo HTML si è evoluto, ed il suo utilizzo ha rapidamente portato a creare altri linguaggi di markup basati su XML per la rappresentazione di particolari informazioni.

Alcuni esempi di altri linguaggi di markup oggi utilizzati sono:

- MathML - Utilizzato per la rappresentazione di formule/simboli matematici
- SVG - Permette la rappresentazione grafica delle immagini
- SMIL - Permette la rappresentazione di presentazioni audio/video

Approfondimenti

- W3C Math Home (MathML) http://www.w3.org/Math/
- Scalable Vector Graphics (SVG) http://www.w3.org/Graphics/SVG/
- Synchronized Multimedia (SMIL) http://www.w3.org/AudioVideo/

2.16 Il W3C

Lo sviluppo del WWW negli anni è stato guidato dal W3C (World Wide Web Consortium). Il World Wide Web Consortium fu creato nell'ottobre 1994 per guidare lo sviluppo del WWW verso il massimo del suo potenziale, sviluppando protocolli comuni che promuovono la sua evoluzione e ne assicurano la sua interoperabiltà. Il W3C ha circa 400 organizzazioni partecipanti sparse in tutto il mondo che collaborano per lo sviluppo del WWW, alcuni dei membri più famosi sono: Adobe, AT&T, Autodesk, Canon, CERN, CISCO, Hewlett Packard, IBM, Intel, Microsoft, SAP, SUN. Il direttore del W3C è Tim Berners-Lee. Le raccomandazioni (Recommendation) del W3C Il W3C ha sviluppato un meccanismo di standardizzazione delle tecnologie del Web basato sulla emissione di raccomandazioni.

La raccomandazione sviluppate dal W3C devono essere formalmente approvate dai membri. Il consenso è raggiunto attraverso un processo di revisioni successive che passa dagli stadi di:

- Working Draft
- Candidate Recommendation
- Proposed Recommendation
- Recommendation[12][1]

Quando una documento raggiunge la fase di Recommendation vuol dire che ha avuto l'approvazione dei membri partecipanti al consorzio ed il processo per diventare una raccomandazione si preoccupa di verificare che la soluzione proposta sia realizzabile (attraverso delle "Call for implementation".

Approfondimenti

- About the World Wide Web Consortium (W3C) http://www.w3.org/Consortium/

[12] La lista delle raccomandazioni è reperibile sul sito del W3C all'indirizzo http://www.w3.org/TR/.

Progettare un sito: parte I

3.1 Capire i media

Progettare un sito web non vuol dire solamente conoscere i linguaggi di markup o i linguaggi di scripting e di programmazione, o addirittura scegliere una immagine di effetto e dei colori attraenti. Sicuramente come riportano Goto e Cotler:

> *"il cliente è ansioso di vedere al più presto la veste grafica, il look-feel del nuovo design", ma, continuano, "l'unica maniera per gettare le basi di un solido sito e per sviluppare efficacemente tutto quanto (incluso il visual design) è pianificare accuratamente la struttura.*[13]*"*

La crescita esponenziale che Internet ha fatto negli ultimi anni, ha senz'altro portato alla nascita di nuove professionalità o all'adattamento di vecchie professioni a quello che sembra essere diventato il nuovo e più diffuso mezzo di comunicazione di massa, il web. Per questo motivo non possiamo non parlare della storia della comunicazione, di cui faremo dei brevi cenni legati all'invenzione dei diversi media. In generale affronteremo il problema della progettazione di un sito web toccando problematiche riguardanti discipline diverse come la linguistica, la psicologia, l'impaginazione grafica, la scrittura per il web, senza tralasciare gli aspetti dell'usabilità e dell'accessibilità di un sito web.

Uno dei problemi fondamentali nella progettazione di un sito web è la sua chiarezza generale. Le domande che un utente si fa più spesso davanti ad un sito web sono sempre le stesse:

- Dove sono?
- Dove posso andare?
- Da dove comincio?
- Dove hanno messo ...?
- Quali sono le cose più importanti nella pagina?
- Perchè l'avranno chiamato così?

Senza dubbio sono domande che riflettono una mancanza di attenzione per gli aspetti di usabilità, ma sicuramente anche di riflessione nella progettazione iniziale del prodotto.

3.1.1 Cosa è un sito web

E soprattutto:

- a cosa serve (informare, divertire, sbalordire, vendere, ...)

[13] Goto, Cotler – Web Redesign – Apogeo 2002

- a chi serve (casalinga, professionista, studente, ...)
- quando serve (a casa, a lavoro, in viaggio, ...)
- ...
- dove serve (sul computer, sul telefonino, sul palmare, ...)

L'importanza di stabilire sin dalle prime fasi della progettazione quali devono essere i requisiti e i potenziali utenti che ne usufruiranno, permette di restringere le possibili alternative nel rispetto delle aspettative finali.

Le immagini sotto riportate mostrano differenze notevoli nel layout delle pagine, nei colori e nell'organizzazione delle informazioni, a seconda della funzione principale che il sito vuole svolgere. Un sito web infatti può servire per:

Fornire informazioni:

Figure 1 – Il catalogo della biblioteca dell'Università di Pisa e la directory di Yahoo

Pubblicizzare un'azienda (company brochure):

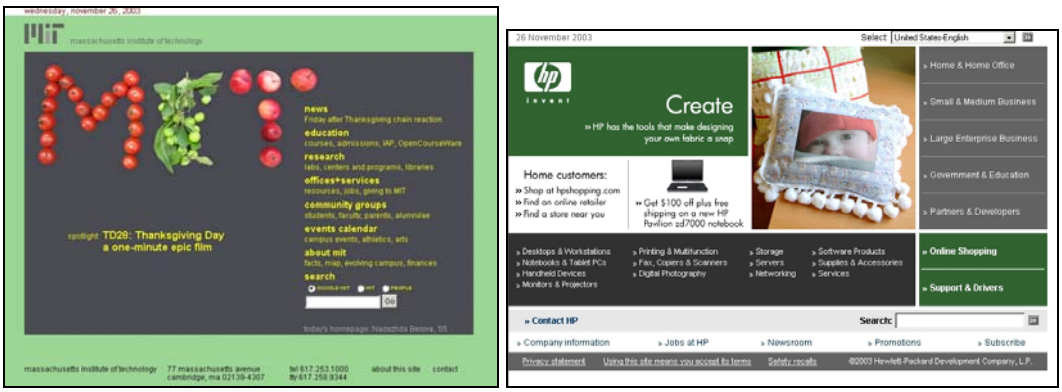


Figure 2 – Il Massachusetts Institute of Technology e la Hewlett Packward

Intrattenere e divertire:

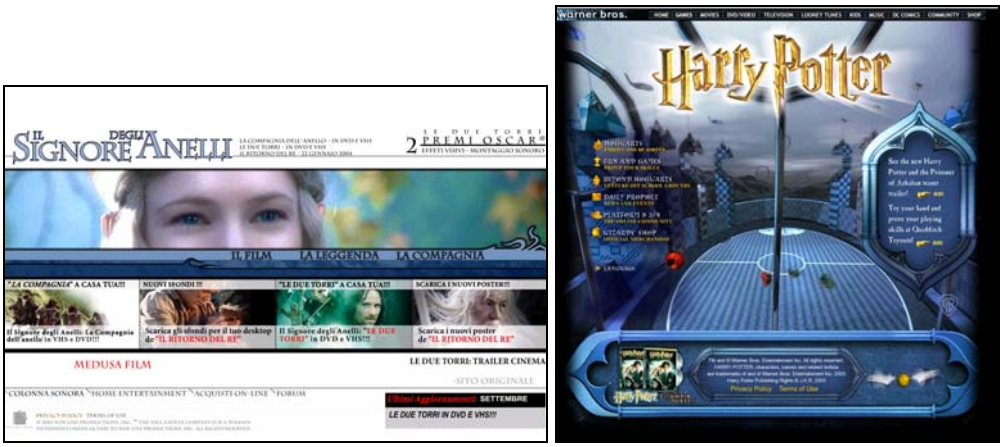


Figure 3 – il Signore degli Anelli e Harry Potter

Vendere:

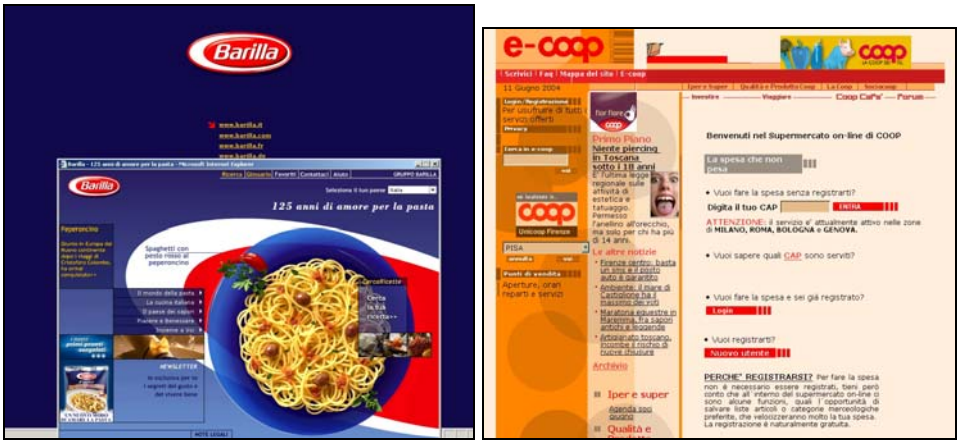


Figure 4 – Barilla e e-coop

Offrire un servizio:

Figure 5 – La Rete Civica del Comune di Pisa e l'URP degli URP

In generale ogni sito web ha le sue esigenze che dovrebbero essere individuate prima ancora di scrivere una singola riga di codice.

Esistono diversi tipi di classificazione per i siti web. Una possibile tassonomia è quella che suddivide i siti in tre grandi categorie, Intranet, Extranet e siti pubblici[14]:

- **siti interni alle aziende (Intranet)**: vengono detti anche intranet o ultimamente EIP (Enterprise Information Portals). Sono quei siti le cui informazioni sono rese disponibili solo al personale interno all'azienda. Non sono quindi accessibili dall'esterno, ma sono in fortissima crescita.
- **siti interaziendali (Extranet)**: sono quei siti le cui informazioni sono rese disponibili solo ai partner dell'azienda, o ai clienti, in ogni caso alle persone autorizzate
- **siti pubblici**: sono tutti quei siti liberamente accessibili via Internet. Sono stati i primi a nascere e ne esistono di diverse tipologie
    - portali (verticali e orizzontali)
    - archivi informatici (biblioteche di testi, di fotografie, musicali, ecc..)
    - giornali
    - siti commerciali
    - brochure aziendali su Internet
    - commercio
    - intrattenimento
    - ...

In ogni caso, qualsiasi tipo di classificazione si tenga in considerazione, la cosa importante è capire obiettivi e funzionalità del sito, che comportano una maggiore o minore percentuale di contenuto a dispetto della interattività, o viceversa, come mostrato nel grafico sotto.

---

[14] Classificazione fornita da Polillo (Etnoteam).

Un'altra classificazione possibile potrebbe essere:

motori di ricerca (search egine - directory)

portali (verticali - orizzontali)

siti commerciali

giornali elettronici

siti divulgativi di carattere generale

siti divulgativi di carattere scientifico

siti vetrina

siti di utenti privati

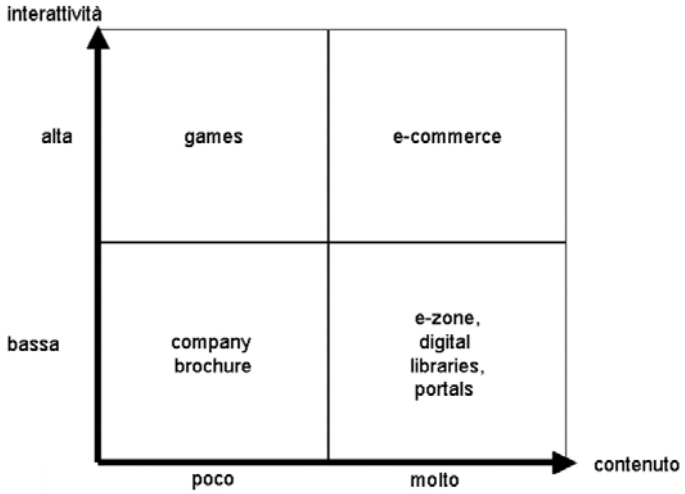


Figure 6 – Classificazione dei siti web in base alla percentuale di contenuto o di interattività (Polillo, Etnoteam)

Il grafico di Figura 6 mostra la relazione esistente fra contenuto e interattività, in tipi diversi di siti web. Sull'asse delle ascisse il contenuto cresce man mano che ci si sposta verso destra, come nelle brochure aziendali, fino ad arrivare alle più ricche librerie digitali. Sull'asse delle ordinate è l'interattività ad aumentare man mano che ci spostiamo verso l'alto, raggiungendo il massimo nei siti di intrattenimento. La massima interattività con il massimo contenuto si ha nei siti di commercio elettronico (ultima casella in alto a destra) dove l'offerta informativa dei prodotti in vendita si accompagna alla complessità delle interazioni delle transazioni online.

Sembra che in un prossimo futuro tutti i siti si spostino verso quest'ultima casella: molto contenuto e alta interattività. Questo fatto viene facilitato dalla sempre maggiore velocità di connessione a Internet e dalle tecnologie in continua evoluzione.

3.1.2 Comunicazione/informazione

Il primo assioma della comunicazione[15] dice che

"non è possibile non comunicare"

Infatti:

- Ogni oggetto del mondo ci appare sensato ... significa.

[15] Il secondo assioma della comunicazione dice che "ogni comunicazione ha un aspetto di contenuto e un aspetto di relazione, di modo che il secondo qualifica il prima ed è quindi Metacomunicazione".

Il terzo assiona della comunicazione dice che "la natura di una relazione dipende dalla punteggiatura delle sequenze di comunicazione dei comunicanti".

Riferimento Paul Watzlawick, Janet Helmick Beavin, Don D. Jackson, Pragmatica della comunicazione umana, Astrolabio, Bologna 1971

- La significazione è la ricchezza di senso.
- La comunicazione si capisce per via di un codice ed è svolta da un emittente.
- La significazione si capisce per via di una regola/abduzione ed è svolta dal destinatario.
- Non vi è comunicazione se non c'è ricezione.
- L'informazione è la riduzione delle incertezze sullo stato del mondo: può essere ridondate, ellittica (o implicita), dipende comunque dal lettore modello e dalle sue aspettative, dal suo bagaglio enciclopedico e culturale (internazionalizzazione), e dall'atteggiamento ideologico.

Anche su web ogni elemento è un elemento comunicativo. Quanto sia efficace la comunicazione dipende da quanto l'utente/destinatario riceve più o meno correttamente l'informazione che abbiamo voluto dare con lo scopo prefisso dal sito. Il messaggio diventa oggetto materiale, e quindi sostituto, dei contenuti mentali che si vogliono trasmettere. Il modo con cui viene attuata tale trasmissione comporta atteggiamenti interpretativi diversi.

Gli elementi in gioco in un processo comunicativo sono quelli mostrati nella figura 7.

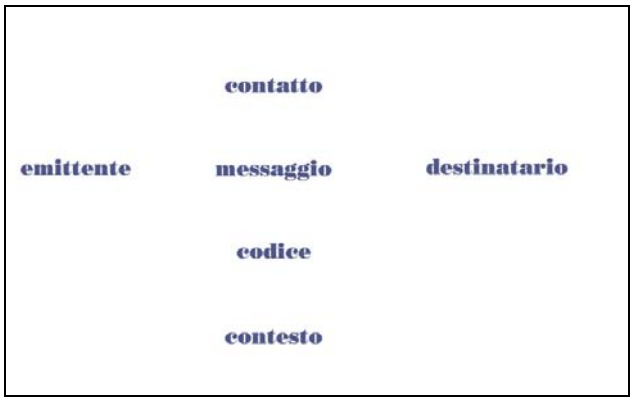


Figure 7 – Elementi della comunicazione

I rapporti fra i diversi elementi determinano le tre dimensioni della comunicazione, dal punto di vista linguistico e semiotico:

- **sintattica:** studia i rapporti tra messaggio/codice/contatto --- organizzazione degli elementi visivi, sintassi del linguaggio
- **semantica:** studia i rapporti tra messaggio/contenuto --- ovvero il modo in cui il messaggio si rapporta con il contesto, quindi con la rete di concetti
- **pragmatica:** studia i rapporti tra messaggio/emittente/destinatario --- quali sono gli effetti e le modalità dell'enunciazione

Ad ogni elemento della comunicazione è poi legata una funzione specifica, come descritto in Figura 8.

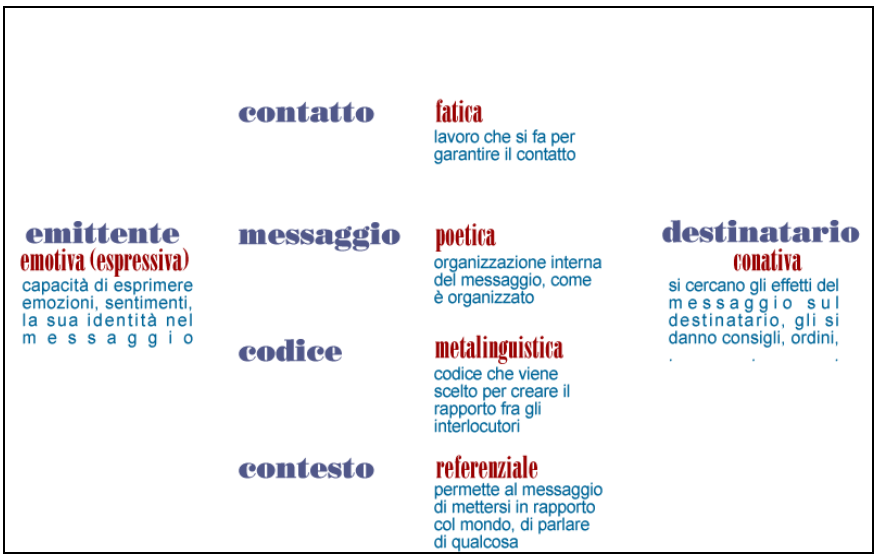


Figure 8 – Le funzioni associate ad ogni elemento della comunicazione

In un processo comunicativo entrano in gioco contemporaneamente più funzioni e mai una sola alla volta. Può anche succedere che qualcuna abbia la prevalenza sulle altre, e ciò dipende essenzialmente dall'enfasi della comunicazione stessa.

3.1.3 Telecomunicazioni e memoria

Lo sviluppo delle telecomunicazioni e delle comunicazioni di massa ha portato sicuramente ad un cambiamento nella memoria della società e di conseguenza nelle memorie individuali.

Con l'introduzione della stampa ed anche con lo sviluppo di sistemi per la memorizzazione, il processo di comunicazione diventa sempre più indipendente dal tempo e dallo spazio.

L'evoluzione dei sistemi di comunicazione ha portato:

- all'evoluzione del supporto, fruibile in tempi e spazi diversi
- allo sviluppo di una comunicazione in "tempo reale" mediata da strumenti informatici
- ad una *"... simulazione della temporalità del dialogo umano"*.

Si ha sempre meno bisogno di una memoria a lungo termine, indispensabile in tempi remoti, così come nelle comunità a cultura orale, mentre si ha sempre più bisogno di una memoria a breve termine. *"Nelle memoria temporanea (a breve termine) si verifica un rapido deterioramento delle informazioni, mentre la memoria a lungo termine conserva le*

informazioni in modo sostanzialmente stabile."[16] L'informazione nella memoria a breve termine viene cancellata molto velocemente per far posto a nuove informazioni. La memoria a breve termine infatti, ha dimensioni limitate; se l'informazione non viene richiamata, più volte, scompare facilmente da quello che viene detto "cuscinetto di ripetizione". L'informazione viene lasciata nel "cuscinetto" fino al suo trasferimento nella memoria a lungo termine o al suo completo ripristino con nuova informazione.

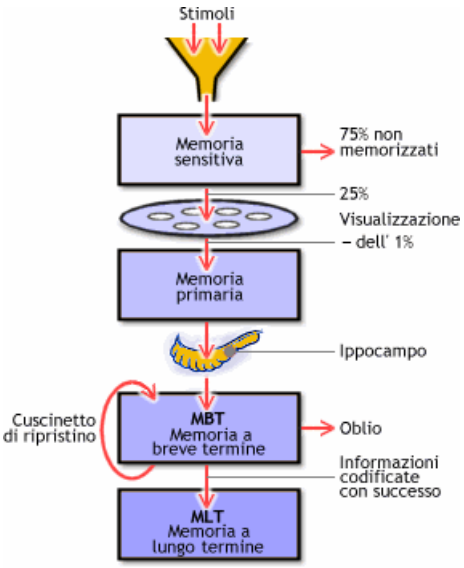


Figure 9. Ciclo della memoria nel cervello umano

Il web ne è un esempio. Gli sforzi cognitivi necessari a capire il da farsi per svolgere un compito devono essere ridotti al minimo ... non ci dovrebbe essere bisogno di un "manuale di istruzioni per l'uso" e non dovrebbe essere necessario studiare il sito prima di poter navigare al suo interno. Questi sono solo alcuni dei concetti alla base dell'usabilità dei siti web, che vedremo in seguito.

La storia delle telecomunicazioni può essere suddivisa in quattro fasi:

- **invenzione del telegrafo**, dove il modello comunicativo è informazionale, ovvero l'importanza è sul segnale che viene trasmesso e non sul contenuto del messaggio
- **invenzione della telefonia:** è simile alla quarta fase (i nuovi media), ma qui si parla di modello comunicativo quasi-interazionale che si differenzia dal modello faccia-a-faccia per la non presenza degli interlocutori nello stesso spazio. Vi è comunque una ulteriore distinzione in questa fase data dalla telefonia fissa (l'apparecchio è legato allo spazio) rispetto alla telefonia mobile (l'apparecchio è legato alla persona).
- **invenzione delle onde-radio (fase radio televisiva):** modello comunicativo diffusivo e monodirezionale. Qui viene simulato il carattere bidirezionale della comunicazione faccia-a-faccia.

[16] http://www.benessere.com/psicologia/arg00/memoria.htm

- **sviluppo dei nuovi media:** modello comunicativo interattivo. Per interattività si intende la comunicazione mediata da un sistema tecnologico, a differenza dell'interazione che si ha nella comunicazione faccia-a-faccia. Nel caso dei nuovi media infatti l'utente non si trova di faccia ad un partner comunicativo, ma "in interfaccia" con esso.

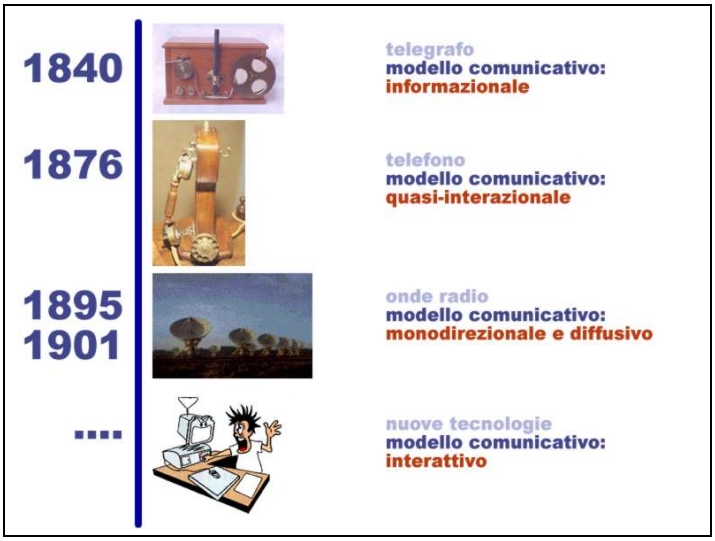


Figure 10 – Le quattro fasi della storia delle telecomunicazioni e i modelli comunicativi collegati a ciascuna di esse

Se è vero che quindi che

"Ogni nuovo progresso accresce i poteri e le capacità della società umana; ogni nuovo progresso moltiplica anche la quantità di conoscenze che le nuove generazioni avrebbero dovuto imparare." [D.Norman - Le cose che ci fanno intelligenti]

è vero anche che è necessario un aggiornamento costante, un adeguamento, un abituarsi ai nuovi mezzi di comunicazione di massa, una **formazione/informazione** continue, sulle potenzialità e sulle possibilità che le nuove tecnologie ci offrono.

In questo contesto il web può essere visto come un modello comunicativo

- **interattivo** (mediazione di strumenti informatici);
- ma anche **quasi-interazionale** (videoconferenze e VoIP[17]);
- e **monodirezionale** (il ricevente non entra in contatto diretto con il ricevente ma con una *"sua istanza simulacrale rappresentata dallo schermo"*[18]).

[17] Voice over Internet Protocol

[18] Giovanni Manetti, L'enunciazione. *Le origini del concetto e alcuni più recenti sviluppi.* Protagon, Siena. 1998

3.2 Web design 1

Definire il progetto è sicuramente la prima cosa da fare. In qualsiasi campo, un buon progetto iniziale riduce i tempi di sviluppo successivi, mentre una cattiva progettazione, o una progettazione troppo affrettata, porta spesso a revisioni in corso d'opera che si rivelano nel complesso molto più costose. La fase di progettazione inziale è quella che porta via molto tempo; per questo motivo troppo spesso viene trascurata o considerata non indispensabile.

3.2.1 La raccolta di informazioni

Nella definizione di un progetto per un sito web uno dei primi passi da compiere è quello della raccolta delle informazioni dal cliente, ovvero da colui che ha commissionato il lavoro. Alcune domande a cui rispondere potrebbero essere del tipo:

- Quali sono i desideri e gli obiettivi del cliente? Qual è il piano operativo per conseguirli?
- Quanto costerà nel suo insieme il progetto, come sono composti i costi e come possono essere ridotti, quante ore vanno allocate ad ogni singolo task?
- Chi sono i membri del team e come si suddividono le rispettive responsabilità?
- Quali sono le responsabilità del cliente?
- Quali materiali occorre produrre nel corso del progetto, e con quali scadenze vanno consegnati?
- Come si verificherà la rispondenza del sito alle esigenze degli utenti?
- Quali sono gli obiettivi a lungo termine del sito?

Esistono questionari già predisposti che possono aiutare a raccogliere le informazioni di cui abbiamo bisogno per iniziare un progetto di design del sito. Sono questionari dettagliati che in qualche modo obbligano il cliente a fornire informazioni utili, a volte date per scontate, ma soprattutto servono allo stesso cliente per chiarirsi meglio intenzioni, obiettivi e requisiti che vorrebbe venissero soddisfatti dal nuovo sito web.

È sicuramente auspicabile crearsi il proprio questionario anche laddove i clienti sono interni all'azienda per cui stiamo lavorando. Sarebbe importante non saltare questa fase che invece viene spesso considerata quasi superflua e "ovvia", ma che aiuta a formalizzare, meglio se per iscritto, il lavoro futuro. Naturalmente alla raccolta dei dati segue un'analisi dettagliata che aiuti a passare alla fase successiva della progettazione.

3.2.2 Il cliente

"I clienti, normalmente, hanno le idee chiare circa gli obiettivi di business ma molto meno chiare circa gli obiettivi del sito." [Goto, Cotler – Web-Redesign – Apogeo 2002]

È importante dialogare con il cliente per capire insieme a lui le esigenze del sito soprattutto dal punto di vista comunicazionale. Per cliente qui intendiamo sia un cliente esterno, nel caso di una agenzia di design, sia un cliente interno all'azienda per cui stiamo lavorando. In ogni caso per cliente viene inteso chiunque, interno o esterno, abbia il ruolo di approvare il progetto, il budget e il contenuto informativo.

Molte informazioni sul cliente possono essere ricavate anche da brochure, cataloghi, vecchie versioni del sito, loghi, ecc.. Tutto ci permette di mettere a fuoco meglio l'idea stessa che il cliente vuole far apparire all'esterno attraverso il modo con cui ha operato fino a quel momento. Non è da escludere il fatto che la necessità di una nuova progettazione del sito web sia dettata da una necessità di nuova immagine verso l'esterno. In questo caso l'operazione di analisi del passato dovrebbe aiutare a creare la nuova immagine, insieme alle informazioni raccolte con il questionario visto in precedenza.

Nel caso in cui si debba partire da zero e non si abbia a disposizione nessun materiale già creato e utilizzato (una nuova azienda, il sito personale, ecc..) è bene dettagliare il più possibile le domande del questionario e organizzare dei colloqui aperti con il responsabile del cliente che ci aiutano a direzionare al meglio l'intervento che stiamo progettando.

3.2.3 L'utente

Non meno importante del cliente è l'utente che visiterà il sito che stiamo progettando. In un paragrafo successivo presenteremo la metodologia User Centered Design (UCD) alla base della progettazione centrata sugli utenti della maggior parte degli artefatti in uso oggi, compreso il sito web.

Laddove possibile, è bene raccogliere il maggior numero di informazioni riguardo all'utenza potenziale del nostro sito:

- quali sono le categorie di utenza che più visitano il nostro sito
- quali sono le loro caratteristiche: background, expertise tecnologico, ecc.
- quali sono le loro esigenze

In questa fase è bene farsi aiutare dallo stesso cliente che molto probabilmente conosce già molte delle informazioni che stiamo cercando. Aiutandosi anche qui con un questionario appositamente strutturato possiamo definire i profili di utenza con le loro competenze e i requisiti

tecnologi più utilizzati: velocità di connessione, piattaforme utilizzate, plug-in, browser, ecc..

Un altro aspetto da non sottovalutare nella definizione della tipologia di utenza del nuovo sito, sono le informazioni di carattere demografico e geografico, quali età, sesso, livello di istruzione, luogo di residenza, ecc... Una cosa infatti è progettare un sito per dei bambini, un'altra è progettarlo per degli specialisti di settore o per un gruppo con specifiche esigenze tematiche.

3.2.4 Gli scenari d'uso

Gli scenari d'uso[19] sono casi rappresentativi delle situazioni reali in cui un utente svolge una attività, dettagliandone le situazioni d'uso. Possono essere rappresentati con fotografie, immagini, descrizioni testuali, storyboard. Aiutano ad immedesimarsi meglio con l'utente che utilizzerà un certo prodotto e quindi ad individuare più velocemente alcune delle problematiche che potrebbero insorgere durante l'interazione.

Esempio: scenario 1

Romina Bianchi ha 29 anni e lavora in un negozio di abbigliamento. Il tempo per lei è molto importante anche perchè ha un bimbo di 1 anno. Vive a Porta a Lucca in un appartamento con 3 stanze. Due o tre giorni alla settimana suo marito è fuori per lavoro e lei vuole mantenersi in contatto con lui.

Vuole inviare le foto di Andrea, suo figlio, a tutti i suoi amici e ai suoi parenti.

Esempio: scenario 2

Aldo ha 56 anni ed è dirigente aziendale da 7 anni. Lavora nel campo dell'abbigliamento ma ha una buona preparazione informatica. Anzi, ha adottato molte tecnologie nella sua azienda ed è convinto della formazione continua del personale, a tutti i livelli.

E' sposato ed ha due figli maschi di 30 e 35 anni: Giorgio e Riccardo. Solo Giorgio vive ancora in casa e gli piace usare il computer che il padre tiene a casa ... è più aggiornato del suo. Spesso Aldo cerca le sue vacanze su Internet e le fa cercare da Annalisa, la moglie. A loro piace andare a trovare i posti più strani in giro per il mondo, ma anche arrangiarsi. Partono sempre informati e preparati sulla cultura e tutto quello che li aspetta. ....

3.2.5 Il re-design

Nel caso di re-design di un sito web, ovvero di ristrutturazione di un sito già esistente, oltre alle domande su cliente, azienda e utente, viste in precedenza, è bene porsi degli interrogativi sugli obiettivi e i motivi di una ristrutturazione. A differenza infatti di un design ex-novo, nella fase di ristrutturazione bisogna fare una ulteriore analisi dell'esistente e capire come migliorare e per cosa.

---

[19] Carroll John M., Scenario-based design: envisioning work and technology in system development, Wiley, Ney York, 1995

Alcuni motivi per il re-design infatti potrebbero essere dovuti ad esigenze di:

- Aumentare il traffico
- Incrementare le vendite
- Migliorare la navigazione
- Facilitare la consultazione
- Impostare una struttura scalabile in vista di una crescita futura
- Creare un sito dinamico orientato ai contenuti

Ciascuna motivazione andrebbe affrontata in maniera diversa, sempre motivandola e cercando di capire quali siano state le problematiche fino a quel momento: una immagine sbagliata, troppi link per compare un prodotto, troppo difficile la transazione online, l'azienda è cresciuta, ecc...

3.2.6 Il project plan

Con la raccolta di tutte le informazioni riguardanti il cliente, l'utente e le necessità di una progettazione/riprogettazione, è bene a questo punto riassumere tutto per iscritto. In questo modo anche a distanza di tempo si ha sott'occhio tutto il quadro delle decisioni prese nella fase preliminare della progettazione, spesso la più importante. È a questo documento che si dovrebbe fare riferimento ogni qual volta si presenta un problema, vuoi con il cliente che cambia opinioni in corso d'opera, vuoi con il team di progettazione che magari decide di seguire una strada diversa da quella definita in partenza.

Nel documento (*project plan*) andrebbero inserite anche le linee scelte per:

- il tono e lo stile del sito (allegro, serio, formale, informale, ...);
- la strategia di comunicazione da adottare, in accordo con il tono e lo stile;
- il messaggio di fondo da fornire;
- la pianificazione dei tempi di sviluppo;
- la definizione del team interno/esterno che si occuperà del progetto, con i rispettivi ruoli e responsabilità.

Purtroppo non sempre è possibile scrivere un documento del genere, spesso per problemi di tempo o di personale dedicato. È vero però che molti problemi in fase di realizzazione e sviluppo del progetto, derivano proprio dalla mancanza di un documento di questo tipo che riporti in dettaglio tutte quelle informazioni stabilite inizialmente con il cliente. Mettere per iscritto quanto discusso oralmente spesso permette di rivedere in maniera più chiara le decisioni prese, riflettendo più approfonditamente prima della loro completa approvazione.

3.3 Visual design

Vediamo ora alcuni degli aspetti grafici e di lettering (scelta dei font e delle loro caratteristiche stilistiche) da tenere in considerazione quando si progetta la parte visuale di un sito web. Ogni supporto mediale ha le sue regole da rispettare e quindi ogni progetto grafico deve adeguarsi a tale esigenze. Nel caso del web la prima cosa da tenere in considerazione è la difficoltà di lettura, sicuramente molto diversa dalla lettura di un libro stampato.

Il visual design in generale non è solo una interfaccia carina, ma creatività grafica, integrazione con gli obiettivi strategici del sito e con le implicazioni funzionali/tecnologiche. Di solito vengono proposti al cliente da uno a tre layout grafici, sufficienti per avere una possibilità di scelta e non creare inutile confusione. Durante la fase creativa il grafico dovrebbe essere sempre in contatto con chi produrrà le pagine per valutare la loro fattibilità tecnica. In questa fase va studiato accuratamente il concept visuale, ovvero gli abbinamenti cromatici in base a: tono comunicativo, mappa del sito, e layout delle pagine.

Nella progettazione del layout grafico delle pagine del sito si deve sicuramente tenere conto di tutte quelle regole tipiche della composizione grafica adattate per il caso specifico del web:

- *logica visiva:* bilanciamento fra le sensazioni visive e le informazioni grafiche;
- *impatto visivo:* necessario a creare un interesse sull'utente ;
- *consistenza:* stabilire un layout che si ripeterà e si riconoscerà in tutte le pagine del sito;
- *dimensioni della pagina:* nel caso di supporti mediali diversi, controllo dello scroll orizzontale (da evitare il più possibile);
- *lunghezza della pagina:* le pagine più lunghe di due scroll verticali completi dovrebbero avere un link per tornare in cima alla pagina.

3.3.1 La lettura su web

La lettura a monitor è più difficoltosa della lettura su un foglio stampato: studi scientifici provano che l'occhio umano impiega il 25% del tempo in più su monitor, rispetto alla stampa.

Molti studi di Eye Tracking, cioè degli spostamenti dell'occhio all'interno di una pagina web, dimostrano che l'utente non legge completamente la pagina come farebbe con un documento stampato, ma piuttosto salta (*scan*) da un

punto all'altro, alla ricerca di un qualcosa che possa avvicinarsi alle sue aspettative.

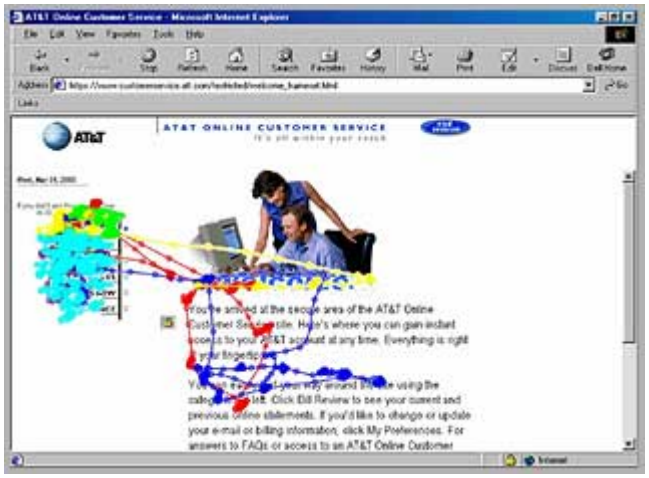


Figure 11 – Spostamenti dell'occhio sulla pagina, evidenziati dalle linee con i nodi (da www. eyetracking.com)

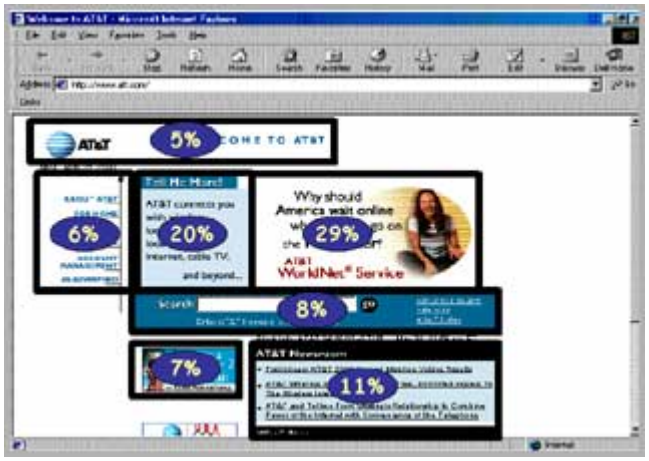


Figure 12 – Percentuale di permanenza dell'occhio sui diversi elementi della pagina (da www. eyetracking.com)

Se le pagine sono troppo piene (di testo, di elementi, ecc.) lo sforzo cognitivo richiesto all'utente può essere eccessivo e provocare l'abbandono del sito. Analogamente si può avere abbandono nel caso in cui i testi sono troppo lunghi, e quindi necessitano dello scroll della pagina, o i paragrafi troppo complicati. Il linguaggio usato deve essere in linea con la scelta comunicativa del sito (scherzoso/serio, formale/informale, ecc..) e non appesantire ulteriormente la lettura con espressioni tecnicistiche o difficili da interpretare, laddove non viene richiesto esplicitamente dalle specifiche di progettazione.

3.3.2 Scelte tipografiche

I tipi di caratteri, *font*, si suddividono in diverse famiglie, ma le due grandi classificazioni che le racchiudono sono:

- *serif*, con grazie
- *sans-serif*, senza grazie

La figura a lato mostra un esempio di carattere con grazie (gli ornamenti cerchiati nella lettera E in alto sono le cosiddette 'grazie'), ed un tipico carattere senza grazie (in basso).

Tutti i sistemi operativi installano per default diversi tipi di font, con grazie e senza grazie, tra cui i più famosi e utilizzati sono:

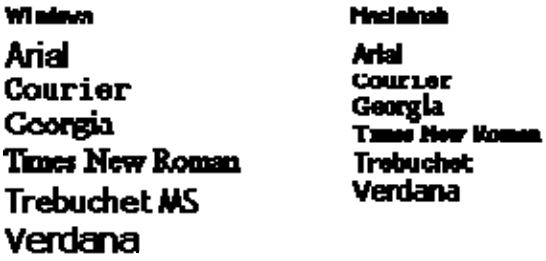


Figure 13 – Alcuni tipi di font installati dai sistemi operative Windows (sinistra) e Macintosh (destra). Arial, Trebuchet e Verdana sono tipici font senza grazie, mentre Courier, georgia e Times New Roman sono i più usati font con grazie.

Tra i sistemi operativi esistono comunque delle differenze di visualizzazione per quanto riguarda le dimensioni dello stesso tipo di font, come evidenziato nella figura 14, per i sistemi operativi Windows e MacIntosh.

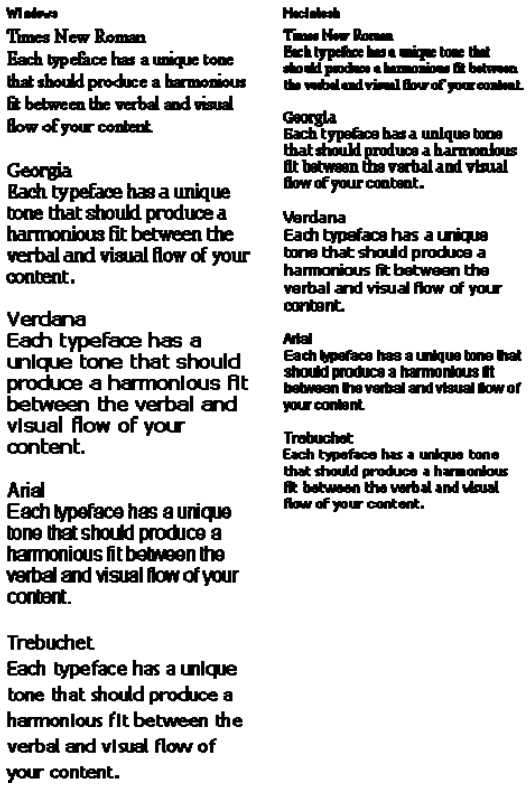


Figure 14 – Lo stesso font, con lo stessa dimensione, viene visualizzato a grandezza diversa a seconda del sistema operative con cui stiamo lavorando.

Le convenzioni classiche per la stampa tipografica normalmente utilizzano un font senza grazie per i titoli, ed un font con le grazie per il testo. Questo è dovuto al fatto che è più facile riconoscere una parola semplicemente scorrendo le grazie di un carattere.

Sul web da anni si dibatte su quale sia la scelta migliore per i titoli brevi e per il testo lungo. Con i primi schermi di computer le scelte erano obbligate dalla bassa risoluzione e venivano quindi utilizzati i caratteri senza grazie per i testi lunghi per non affaticare ulteriormente la lettura, già lenta. I monitor a bassa risoluzione, infatti, non permettevano una chiara leggibilità delle grazie dei testi scritti per esempio con un font come il Times. Oggi con l'evoluzione delle tecnologie utilizzate per la costruzione dei monitor e con l'elevata risoluzione con cui riusciamo a visualizzare, è possibile dire che la differenza nell'utilizzo di un tipo di font rispetto ad un altro non cambia molto le prestazioni nella leggibilità.

La maggior parte dei font può avere caratteristiche diverse, come:

- il **grassetto**
- il *corsivo*
- il ***grassetto corsivo***
- il <u>sottolineato</u>

L'insieme delle caratteristiche stilistiche di un font, aiutano ad evidenziare una parte di testo e renderlo più importante e visibile rispetto al testo normale. È bene non approffittare troppo delle caratteristiche stilistiche di

un font per non cadere nell'effetto contrario, e probabilmente non voluto, di non far risaltare niente.

Una delle convenzioni ormai diffuse e adottate dagli sviluppatori e dagli stessi utenti del web, è quella di identificare un testo sottolineato come un link da poter seguire. È quindi buona norma non utilizzare questa caratteristica stilistica per evidenziare una parte di testo, che verrebbe molto probabilmente confusa con un link attivo. Possiamo sicuramente trovare metodi alternativi per focalizzare l'attenzione su parti di testo.

Alcuni utilizzano del testo lampeggiante per attirare l'attenzione dell'utente. In questo il problema è che spesso il testo del messaggio è molto lungo e la difficoltà di lettura enorme. E' stato verificato che in persone con particolari problemi cognitivi si potrebbero verificare anche effetti indesiderati quali attacchi epilettici. Buona norma è cercare di usare il testo lampeggiante il meno possibile, meglio mai, e se proprio non ne possiamo fare a meno, per testi brevissimi di al massimo una o due parole.

Altre caratteristiche di stile di un testo sono il maiuscolo, il minuscolo il maisucoletto. Uno degli errori più ricorrenti è quello di scrivere delle porzioni di testo tutte in carattere maiuscolo, credendo erroneamente di attirare l'attenzione. Purtroppo l'occhio umano fa difficoltà a leggere i caratteri maiuscoli rallentando anche la comprensione stessa del testo. La figura 15 mostra a sinistra il caso di parole tutte scritte con lettere maiuscole racchiuse da quello che viene definito un "rettangolo monotono". Sulla destra le differenze evidenti nelle altezze dei caratteri con ascendenti o discendenti, comportano un movimento oculare che aiuta la leggibilità.

Figure 15 – Differenze nella lettura del carattere maiuscolo o minuscolo, dato dallo spazio che circoscrive la parola.

Il colore del testo è un altro elemento fondamentale nella composizione di una pagina web e tiene conto anche di altri aspetti come lo sfondo su cui è inserito, gli elementi stessi che compongono l'intera pagina web, problematica di accessibilità.

Nella scelta dei colori da utilizzare per il testo (sia esso testo lungo che titoli) è bene tenere in considerazione alcune importanti premesse:

- il contrasto tra testo e sfondo deve essere evidente. Per ovviare ad errori banali di contrasto esistono dei tool appositi che aiutano nella visualizzazione e quindi nella scelta dei colori migliori, o addirittura una formula che permette di stabilire il miglior contrasto per la lettura;
- il colore può essere usato per evidenziare un testo, ma non DEVE essere l'unico modo per veicolare una informazione importante (vedi

capitolo sull'accessibilità);esistono delle convenzioni adottate da molti su web e che

- riguardano i colori dei link (attivi, visitati, ecc..). Alcuni affermano che i link devono avere i colori standard che tutti conoscono (viola,ecc.) e devono essere sottolineati. Nel caso di una buona progettazione, una chiara navigazione può essere evidente anche con l'utilizzo di altri colori e senza sottolineatura.
- il cambio di colore al passaggio del mouse aiuta sicuramente ad attirare l'attenzione sulla funzionalità del testo/link attivo. Allo stesso modo un cambiamento nella dimensione del carattere con cui è stato scritto il testo del link, mentre ci passiamo sopra con il mouse, può aiutare di più una certa categoria di utenti con particolari problemi (ipovedenti, per esempio).
- il cambio di colore dopo aver visitato un link attivo aiuta soprattutto nell'orientamento durante la navigazione: dove sono stato.

3.3.3 Layout di pagina

Il layout generale della pagina è uno degli aspetti fondamentali per veicolare un certo tipo di messaggio, ma soprattutto per dare ordine e chiarezza nella distribuzione degli elementi contenuti nella pagina stessa.

La cosa fondamentale quando si pensa al layout di una pagina web, è quella di separare la struttura (il layout vero e proprio con tutte le sue caratteristiche di allineamento, scelta di font, colori, ecc..) dal contenuto (ovvero il testo vero e proprio). Come vedremo meglio in seguito, separare la struttura della pagina dal suo contenuto, oltre a dare una maggiore strutturazione al contenuto stesso (stabilendo con chiarezza i diversi livelli dei titoli, le spaziature tra il testo, le interlinee, ecc..) aiuta a migliorare l'accessibilità del sito.

Figure 16 – A sinistra una pagina con gli elementi distribuiti in maniera casuale all'interno dello spazio disponibile; a destra una organizzazione molto più chiara e pulita

Figure 17 – Allineamento del testo ripsettivamente a sinistra, a destra, al centro e giustificato

Di seguito vengono mostrati alcuni esempi di strutturazione dell'informazione e leggibilità del testo, ripresi da Robin Williams, The non-designer's design book: design and typographic principles for the visual novice[20].

Esempio 1: strutturare l'informazione per una migliore leggibilità delle informazioni

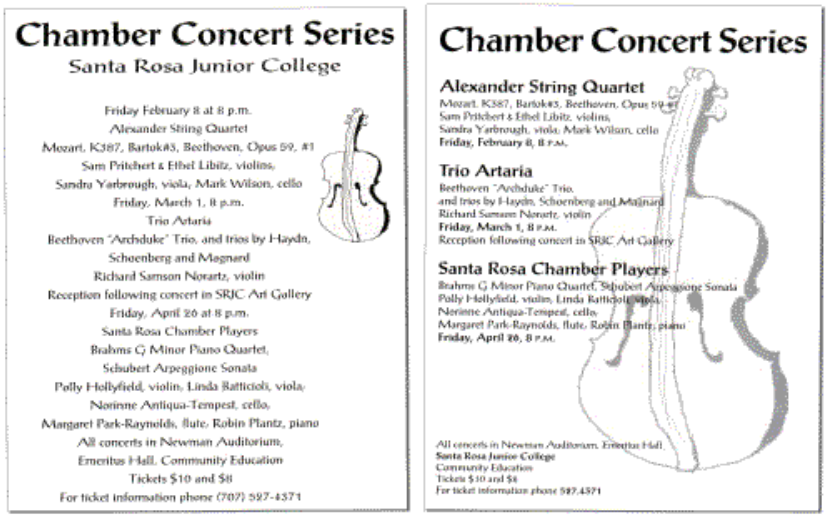


Figure 18 - Raggruppare elementi semanticamente contigui attraverso la vicinanza spaziale

[20] Robin Williams, The non-designer's design book: design and typographic principles for the visual novice, Peachpit Press, Berkeley, 1994

Figure 19 - Più sono vicini fra loro due o più elementi, maggiore è la probabilità che essi siano visti come una figura.

Esempio 2: evidenziare parti di testo

Figure 20 – Usare il contrasto per aggiungere interesse visuale al documento e connessioni fra gli elementi

Introduzione all'HTML

4.1 Cosa è l'HTML?

L'HTML è la lingua del World Wide Web. È un linguaggio di markup **non proprietario** basato su SGML.

L'HTML può essere generato ed elaborato in modi molto diversi: dai semplici editor testuali, disponibili su tutti i computer, agli strumenti WYSIWYG (What You See Is What You Get) molto sofisticati che permettono di creare pagine HTML, e non solo, consentendo a chi li usa di puntare l'attenzione sulla realizzazione della grafica e dei contenuti ignorando – o almeno tentando di ignorare – il codice sottostante. Un altro modo per generare HTML è quello di usare un generatore dinamico di codice a livello server, come ad esempio CGI, JSP, eccetera.

L'XHTML *(Extensible Hypertext Markup Language)* è nato per riprodurre ed estendere l'HTML, riscrivendolo in XML. Nell'ottica del W3C indica le nuove versioni dei linguaggi di markup per il WEB e quindi introduce le nuove tecnologie XML ad esso associabili (Mathml, SVG, ...).

L'XML *(Extensible Markup Language)* è un linguaggio di markup nato per l'editoria elettronica, ma diffusosi largamente anche per lo scambio di dati di natura molto diversa. La sua caratteristica principale è quella di essere semplice e generico, prendendo spunto anche dal sistema di markup HTML.

L'ultima raccomandazione del W3C per gli ipertesti sul web è XHTML 1.1 (dal maggio del 2001).

In questo testo citeremo l'HTML sottointendendo l'XHTML se non specificato diversamente.

4.2 Struttura del documento

Un documento HTML è composto dal **testo** e dagli **elementi**, che ne definiscono la struttura. Ciascun elemento è individuato dai **tag**.

Ecco un esempio di documento HTML:

```
<html>
 <head>
  <title>Titolo</title>
 </head>
 <body>
  corpo del documento
 </body>
</html>
```

4.2.1 I tag

In HTML un **tag** (etichetta) è composto da una *parola* seguita opzionalmente da una lista di attributi, il tutto racchiuso tra i caratteri `<` e `>`.

Una *parola* è formata da caratteri alfanumerici: dalla 'a' alla 'z' e dallo 'o' al '9'. I tag HTML possono essere

- di apertura:

```
<parola>
```

- di chiusura:

```
</parola>
```

- di apertura e chiusura:

```
<parola />
```

4.2.2 Gli elementi

Un elemento HTML è normalmente delimitato da un tag aperto e uno chiuso, come nell'esempio seguente:

```
<h1>
 corso di tecnologie web
</h1>
```

In questo caso si indica il testo soprastante con elemento `h1`.

All'interno dei tag di apertura e chiusura possono essere presenti altri elementi o del testo (il *corso di tecnologie web* dell'esempio sopra).

La serie di *elementi* forma la struttura del documento HTML.

4.2.3 Gli elementi in XHTML

Una differenza tra HTML ed XHTML è che ogni tag aperto **deve** essere chiuso *(da XML, well formed)*. L'esempio seguente indica una corretta chiusura degli elementi XHTML:

```
<title>
 Il titolo
</title>

<p>
 testo di un <strong>paragrafo</strong>
</p>
```

In XML (e quindi anche in XHTML) si è preferita la prolissità alla semplificazione delle prime versioni di HTML. Ogni parte della struttura deve essere dichiarata in modo esplicito e non ambiguo.

4.2.4 Elementi vuoti

Alcuni elementi non hanno tag di apertura e di chiusura separati. Ciò è possibile quando tutta l'informazione è racchiusa nell'elemento stesso e nei suoi attributi e quando l'elemento stesso non ha elementi contenuti (figli).

Esempi di elementi vuoti:

```
<br />

<input type="hidden" name="nome" value="prova" />
```

Nota: in questo tipo di tag di apertura e chiusura è buona norma introdurre uno spazio prima della sequenza di caratteri `/>`, ciò per rendere il codice compatibile con alcune versioni di browser precedenti all'XHTML (in particolare Netscape4).

4.2.5 I caratteri di spaziatura

I caratteri di spaziatura (fine riga, tabulazioni e spazi) anche se ripetuti, vengono normalmente trasformati in singoli spazi durante la presentazione.

Esempio (sorgente HTML e Risultato in un browser tradizionale):

```
Tanto va la    gatta al
    lardo che ci
  lascia lo
    zampino.
```

Tanto va la gatta al lardo che ci lascia lo zampino.

Nota: sui browser testuali o grafici la frase può anche essere spezzata su più righe, in base alle necessità di impaginazione.

4.2.6 Gli attributi

Gli attributi descrivono le proprietà degli elementi. La lista degli attributi è composta da una serie di `nomeattributo="valore"` separati da caratteri di spaziatura ed è innestata nel tag di apertura o in quello di apertura-chiusura:

Esempi di tag con attributi al seguito:

```
<elemento1 attributo1="valore1" attribute2="valore2">
 contenuto
</elemento1>

<h1 class="principale">
 titolo
</h1>
```

Il valore degli attributi viene mantenuto inalterato (nell'esempio sopra la lista dei valori è: valore1, valore2, principale), può essere maiuscolo o minuscolo e può contene contenere uno o più spazi anch'essi lasciati inalterati.

Esempio di tag con attributi diversi

```
<h2 title="il titolo">
```

```
<h2 title="IlTitolo">
```

L'ordine con cui sono riportati gli attributi all'interno della lista degli attributi nel tag di apertura (o apertura-chiusura) non è importante.

Esempi di tag con attributi considerati uguali

```
<img src="sorriso.png"
 alt="Sorriso" />
```

```
<img alt="Sorriso"
 src="sorriso.png" />
```

4.2.7 Differenze tra HTML e XHTML

Tra HTML e XHTML ci sono due importanti differenze riguardanti l'interpretazione e l'uso degli attributi e dei tag.

1. In XHTML il valore degli attributi deve essere racchiuso dal carattere " (doppio apice), mentre in HTML non era necessario.
2. In XHTML i nomi degli elementi e degli attributi devono essere scritti in minuscolo, mentre nelle specifiche HTML fino alla versione 4.01 si permetteva anche la forma maiuscola.

Entrambe la differenze si rendono necessarie per compatibilità con XML.

XHTML non valido, HTML4.01 valido	*valido comunque*
`<H1 CLASS=titolone>titolo</H1>`	`<h1 class="titolone">titolo</h1>`

4.2.8 Annidamento

Gli elementi HTML possono essere innestati l'uno dentro l'altro in modo da comporre una struttura gerarchica. Comunque si deve sempre chiudere l'ultimo elemento aperto, prima di poterne chiudere altri. Ecco un esempio di annidamento corretto e scorretto:

corretto	*errato*

`<p>`
`  <strong>testo</strong>`
`</p>`
	`<p>`
	`  <strong>testo`
	`  </p>`
	`</strong>`

Nota: come nell'esempio *corretto* sopra riportato è buona norma usare l'indentazione il testo in base ai tag ed al loro annidamento: questo per dare maggior chiarezza e leggibilità alla struttura del testo, anche se ovviamente il browser comunque interpreta il codice scritto..

4.3 Codifiche dei caratteri

Per far sì che il browser interpreti correttamente il testo presente nel documento HTML è necessario indicare la codifica dei caratteri che verranno utilizzati per effettuare la stesura del documento. In effetti il browser deve interpretare ogni carattere contenuto nel documento attraverso una codifica. Le codifiche consentono l'interpretazione dei valori binari (i byte appunto). Con un byte si possono rappresentare al massimo 256 simboli e quindi, decidendo di associare un carattere ad uno dei possibili 256 valori si ottengono un massimo di 256 caratteri (codifiche a 8 bit = 1 byte).

Per codificare più di 256 simboli, si devono poter usare più byte per rappresentare un singolo carattere.

4.3.1 ASCII

Una delle prime codifiche usate sui computer, e forse la più semplice, è la codifica ASCII, che è composta da 128 caratteri (7 bit) ed include:

- il normale alfabeto inglese sia maiuscolo che minuscolo (`a-z`, `A-Z`)
- i caratteri numerici (`0-9`)
- i caratteri di punteggiatura (`,.:-+/ `)
- alcuni simboli e caratteri di controllo (tab, "a capo", ...)

Esempio di codice ASCII

simbolo	*numero*
A	65
z	122
0	48
9	57
-	45

4.3.2 ISO-8859 e altre codifiche

Tra le codifiche più usate ci sono la serie delle codifiche ISO-8859 composte da 256 caratteri.

La codifica comunemente usata sul nostro Web è quella ISO-8859-1, detta Europea occidentale, la quale estende la codifica ASCII e introduce alcuni simboli aggiuntivi, quali ad esempio le lettere accentate (è, é, ì, ...), sui restanti 128 simboli.

Recentemente è stata introdotta la codifica ISO-8859-15, che basandosi sulla ISO-8859-1, ne sostituisce alcuni simboli poco usati in Europa, con altri tra cui l'euro *(€)*.

Nell'Europa dell'est va per la maggiore la codifica cirillica, ISO-8859-5.

Altre codifiche come quelle asiatiche hanno bisogno normalmente anche di **font supplementari** installati sul computer client, che verranno usati dai browser per presentare correttamente i simboli della codifica.

4.4 Codifica del documento

Nei documenti XML (e quindi anche XHTML) e HTML 4.0 la codifica interna (anche ai browser) di riferimento è Unicode.

Unicode è uno standard creato per affiancare la specifica ISO/IEC-10646 con la quale si definisce un insieme di caratteri universale (UCS).

Con UCS si assegna un numero univoco ad ogni caratteri del mondo; per adesso sono circa 50000 simboli diversi. Lo standard Unicode è definito dall'Unicode Consortium e viene sviluppato in modo sincronizzato assieme allo ISO/IEC 10646. In pratica è un sovrainsieme di qualsiasi altro insieme di caratteri. Per lo standard Unicode l'IETF raccomanda l'uso della codifica **UTF-8**; per maggiori informazioni si veda la RFC 2279 (http://www.ietf.org/rfc/rfc2279.txt)

Per leggere documenti XML e (X)HTML si deve sapere quale codifica dei caratteri usare per poterne interpretare correttamente il contenuto. Questo può essere fatto in alcuni modi:

- nell'intestazione *Content-Type* del protocollo HTTP per mezzo del parametro charset:

```
Content-Type: text/html; charset=utf-8
```

- per i documenti XML (e quindi XHTML) si usa lo pseudo-attributo encoding nella dichiarazione xml all'inizio del documento:

```
<?xml version="1.0" encoding="utf-8" ?>
```

- per i documenti HTML si usa l'elemento meta all'interno del tag `head`:

```
<meta http-equiv="Content-Type" content="text/html;
charset=utf-8" />
```

Nota: si parlerà dell'elemento *meta* più avanti.

È **molto importante** che nei documenti XML e HTML sia indicata la codifica dei caratteri usata, altrimenti il browser deve fare delle assunzioni, magari usando una impostazione predefinita, e di conseguenza può sbagliare. È altrettanto importante che le indicazioni per la codifica della pagina, date dalla pagina HTML stessa e dal server che la fornisce (viste poco sopra), siano coerenti, altrimenti non ne è garantita la corretta interpretazione.

4.4.1 Caratteri speciali (entità)

Per inserire nella pagina HTML dei caratteri non presenti nella codifica del documento si possono usare due diverse sintassi.

Nella prima che prendiamo in considerazione il carattere viene indicato per mezzo del codice numerico Unicode corrispondente con la sintassi **&#NNN;**

Esempio:
© indica il simbolo ©

Nell'altra sintassi, da preferire per la maggiore espressività sia per chi scrive che per chi legge il codice HTML, si indica il simbolo direttamente con il suo nome, cioè si scrive **&nomesimbolo;**

In alcuni casi è necessario usare la sintassi per i caratteri speciali, perché i simboli sono componenti del sistema di markup. In particolare i caratteri **<** e **>** sono usati per delimitare i *tag* e anche se sono presenti nella codifica ASCII, è necessario usare le entità rispettivamente *<* e *>*.

Alcuni simboli:

nome	*simbolo*
€	€
©	©
µ	µ
α	α
β	β

Il riferimento per i possibili nomi di caratteri sono:

- Caratteri Latin-1: http://www.w3.org/TR/xhtml1/DTD/xhtml-lat1.ent
- Caratteri speciali: http://www.w3.org/TR/xhtml1/DTD/xhtml-special.ent
- Simboli: http://www.w3.org/TR/xhtml1/DTD/xhtml-symbol.ent

Il riferimento per i simboli e i codici numerici è:

sgml entities http://www.w3.org/TR/html4/sgml/entities.html#h-24.3

Nota: non tutti i caratteri vengono rappresentati dai browser, conviene provare quelli meno consueti.

4.4.2 Spazi che uniscono: nbsp

Il simbolo ` ` (no breaking space) è uno spazio che separa due parole, ma impedisce che le parole siano separate da una riga nella presentazione.

Esempio senza e con nbsp:

Sorgente	*Risultato*
`Paperon de Paperoni`	Paperon de Paperoni
`Paperon de Paperoni`	Paperon de Paperoni

Nota: in moltissime pagine web questo simbolo viene usato per scopi di formattazione e questo è **sbagliato e come effetto collaterale fa aumentare la dimensione della pagina**. Ci sono proprietà CSS che possono essere usate per avere lo stesso comportamento nella presentazione, ma non aggiungono un contenuto semantico ("de Paperoni" è il cognome, quindi un unica entità).

Struttura di una pagina

5.1 Struttura del documento HTML

Questo capitolo contiene la descrizione della struttura di un documento HTML e dei principali elementi utilizzati per la realizzazione e la formattazione del testo in esso contenuto.

Un documento HTML è composto da due parti principali:

1. Due linee, che definiscono il **tipo di documento** (versione dell'HTML e caratteri utilizzati);
2. Il **documento HTML** che a sua volta è suddiviso in:
   2.1. una sezione dichiarativa, chiamata **intestazione**;
   2.2. il **corpo**, che racchiude il contenuto del documento. Il corpo può essere realizzato utilizzando gli elementi **body** o **frameset** (nel caso in cui siano utilizzati i frame per la strutturazione della pagina).

Esempio:

```
<?xml version="1.0" encoding="UTF-8"?>     } Definizione del tipo
<!DOCTYPE html ... >                       } di documento

<html>

<head> ... </head>  <-- Intestazione del documento } Documento
<body> ... </body>  <-- Corpo del documento       }

</html>
```

5.1.1 Definizione del tipo di documento

Un documento XHTML non è valido e può non essere correttamente interpretato dal Browser se non contiene nelle prime due righe:

- Prima riga: una direttiva XML che dichiara:

a. la versione dello standard XML sulla quale si basa il documento;
b. la codifica dei caratteri utilizzata per definire il contenuto del documento.

- Seconda riga: la dichiarazione della versione dell'HTML utilizzato nel documento - comando SGML contenente la DTD (Document Type Definition).

Definizione del tipo di documento: prima riga

Per descrivere la direttiva XML contenuta nella prima riga del documento utilizziamo un esempio:

```
<?xml version="1.0" encoding="UTF-8"?>
```

In questo caso,

- l'attributo `version="1.0"` dichiara che il documento è un documento XHTML e identifica la versione a cui fa riferimento[21];
- l'attributo `encoding="UTF-8"` precisa che la codifica del carattere utilizzata dal documento è UTF8, che è la codifica di default.

Da notare che nella versione HTML 4.01 e precedenti questa direttiva non era obbligatoria poiché non erano documenti XML.

Definizione del tipo di documento: seconda riga

La seconda riga contiene la dichiarazione `doctype` utilizzata per indicare la DTD usata da un documento XML; la dichiarazione `doctype` deve comparire prima del primo elemento del documento[22].

La DTD è una guida che

a. identifica gli elementi del linguaggio;
b. specifica gli attributi che si possono utilizzare;
c. regola l'utilizzo combinato degli elementi;
d. identifica le informazioni contenute negli elementi o cosa gli elementi rappresentano.

I tre "dialetti" dell'XHTML sono contenuti in tre DTD diverse che definiscono l'XHTML Strict, Transictional e Frameset.

---

[21] Al momento la sola versione completa dell'XML è la 1.0

[22] I browser non utilizzano la DTD specificata nel documento. Le informazioni sulla definizione del tipo di documento sono di solito utilizzate dai validatori.

Di seguito sono indicate le tre diverse dichiarazioni **doctype** da utilizzare per ogni "dialetto".

1. **XHTML 1.0 Strict DTD,** che include tutti i tag e gli attributi non deprecati o relativi ai frame:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

2. **XHTML 1.0 Transitional DTD,** che include tutti i tag e gli attributi contenuti nell'HTML Strict più quelli deprecati

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

3. **XHTML 1.0 Frameset DTD,** che include tutti i tag e gli attributi dell'HTML Transitional più quelli relativi ai frame

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
    "http://www.w3.org/TR/xhtml1/xhtml1-frameset.dtd">
```

All'interno di ogni dichiarazione:

- **PUBLIC,** indica che la DTD utilizzata è pubblica, cioè registrata con un identificativo e resa pubblica dal W3C – nel caso si voglia utilizzare una DTD privata deve essere utilizzata la parola chiave **SYSTEM** invece di **PUBLIC**;
- **-//W3C//DTD XHTML ..... //EN** è l'identificatore della DTD, necessario nel caso in cui la DTD sia pubblica; l'identificatore da modo all'elaboratore XML di individuare la DTD che non sia in una posizione specifica;
- **http://www.w3.org/TR/xhtml1/DTD/xhtml1-.....dtd** è la URI che punta ad una copia della DTD indicata nella dichiarazione.

Esempio: (dichiarazione di un documento che utilizza l'XHTML 1.0 Transitional)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>
 ...
 ...
</html>
```

Documento

5.1.2 Il documento HTML

`<html> … </html>`

Il documento HTML è delimitato dai tag di apertura e di chiusura **`<html>`** e **`</html>`** che contengono tutto il testo e gli elementi di struttura del documento stesso; tutto quello che sta fuori del documento è ignorato dai Browser.

L'elemento **`html`** è l'elemento principale ed è il primo dopo la dichiarazione **`DOCTYPE`**.

Attributi:

- **`lang`, `xml:lang`**, specificano le informazioni sulla lingua utilizzata nell'elemento e nei suoi attributi. I possibili valori dell'attributo sono specificati nell'RFC1766.

Gli attributi **`lang`** e **`xml:lang`** sono attributi comuni a tutti gli elementi dell'HTML ma l'elemento `html` è un buon esempio in quanto da la possibilità di definire la lingua per tutto il documento[23].

Da notare che, l'attributo **`xml:lang`** sarebbe quello più corretto da utilizzare ma, dal momento che alcuni client non lo interpretano ancora, è buona norma metterli tutti e due.

[23] Questo è vero perché l'attributo `lang` è ereditato da tutti i figli; la regola vale per lui e per tutto quello che è contenuto.

Esempio:

```
...                                          ← Definizione del
                                               tipo di documento

<html lang="it" xml:lang="it">
  <head> ... </head>          ← Intestazione   } Documento
  <body> ... </body>          ← Corpo          }
</html>
```

Le informazioni sulla lingua sono utili ai motori di ricerca, ai sintetizzatori vocali, ai controllori di ortografia e grammatica, ecc.

5.1.3 Intestazione del documento ed elementi contenuti

<head> … </head>

L'intestazione del documento è definita dall'elemento **<head> ... </head>** che è inserito tra i tag di apertura **<html>** e **<body>**.

L'elemento **head** contiene una serie di elementi che descrivono le proprietà del documento come il titolo, la posizione all'interno del Web e le relazioni che ha il documento stesso con altri documenti. L'elemento **head** contiene, inoltre, altri elementi utili per la definizione e la gestione del contenuto del documento come le proprietà dei CSS per la presentazione del documento e la definizione di script chiamati all'interno del documento stesso.

Tra gli elementi contenuti nell'elemento **head**, in questo documento tratteremo quelli più importanti: **title**, **base**, **link**, **script**, **style**.

Esempio:

```
...                                    ← Definizione del tipo di documento

<html>
<head>                                 ⎫
  <title> ... </title>                 ⎪
  <base ... />                         ⎪
  <link ... />                         ⎬ Intestazione del Documento
  <script> ... </script>               ⎪
  <style> ... <style>                  ⎪
  …                                    ⎭
</head>
<body> ... </body>                     ← Corpo del Documento
</html>
```

5.1.3.1 Titolo del documento

<title> … </title>

Il titolo del documento è definito dal testo, non formattato, inserito tra i due tag di apertura e di chiusura **<title>** e **</title>**.

title è il solo elemento obbligatorio all'interno dell'elemento **head**.

La scelta del titolo è fondamentale per l'utilizzo del documento nel Web. Il titolo, infatti, deve identificare il documento sia in riferimento ad altri documenti contenuti, ad esempio, nello stesso sito web, sia al suo contenuto. Il titolo è utilizzato dai Browser che lo inseriscono come titolo della finestra e spesso diventa il nome del collegamento al documento quando viene inserito nei link preferiti o nei bookmarks dell'utente.

Esempio:

```
...
<html>
<head>
  <title> Questo è il titolo del documento </title>
  ...
</head>

<body> ... </body>
</html>
```

5.1.3.2 Posizione del documento all'interno del Web

`<base />`

L'elemento **`base`**, inserito all'interno dell'elemento **`head`**, specifica la URI assoluta sulla quale si basano i collegamenti relativi inseriti nel documento[24].

Esempio:

```
<html>
<head>
  ...
  <base href="http://www.nic.it/RA/index.html" />
</head>
<body>
  ...
  <a href="../PM/domain_form.html"> domain form </a>
  ...
</body>
</html>
```

[24] Parleremo delle URI assolute e relative più avanti nel paragrafo 7.1 '*Collegamenti ipertestuali*'

Nell'esempio sopra riportato, la URI relativa `"../PM/domain_form.html"` sarà risolta dal Browser come `http://www.nic.it/PM/domain_form.html`

Se l'elemento **base** non è specificato, si considera il valore relativo impostato automaticamente alla pagina attuale.

5.1.3.3 Relazioni con gli altri documenti

<link />

L'elemento **link** definisce la relazione che c'è tra il documento stesso ed un altro documento nel Web.

Attributi:

- **href**, specifica la URI del documento correlato;
- **type** contiene il 'content type' del documento correlato in formato MIME (opzionale)
  esempio: **text/css**, **text/plain**
- **rel**, **rev**, specificano la relazione tra il documento corrente e il correlato e vice-versa;
  esempio: **next**, **prev**, **stylesheet**

Di seguito è indicata una lista di valori per gli attributi **rel** e **rev**:

Stylesheet

Indica uno style sheet esterno. È spesso utilizzato insieme con l'elemento link di tipo "Alternate" per permettere di selezionare uno style sheet alternativo.

Next

In una sequenza di documenti, indica il documento successivo. Può essere utilizzato dai client per pre-caricare il documento successivo al fine di ridurre i tempi di attesa.

Prev

In una sequenza di documenti, indica il documento precedente. Alcuni client supportano il sinonimo "Previous".

Start

Indica il primo documento di una collezione. È utile per indicare ai motori di ricerca qual'è, secondo l'autore, il punto di partenza di un insieme di documenti.

Alternate

Indica una versione sostitutiva del documento in cui è contenuto l'elemento **link**. Se utilizzato insieme all'attributo **'lang'**,

indica una versione tradotta del documento. Se utilizzato insieme all'attributo `'media'`, indica una versione del documento progettata per essere interpretata da un diverso device.

`Contents`

Indica il documento utilizzato come "sommario" nel documento corrente. Alcuni client supportano il sinonimo ToC (da "Table of Contents").

`Index`

Indica il documento utilizzato nel documento corrente come indice.

`Glossary`

Indica il documento contenente un glossario di termini utilizzati nel documento corrente .

`Copyright`

Indica il copyright.

`Chapter`

Indica un documento indicato come capitolo in una collezione di documenti.

`Section`

Indica un documento indicato come sezione in una collezione di documenti.

`Subsection`

Indica un documento indicato come sotto-sezione in una collezione di documenti.

`Appendix`

Indica un documento utilizzato come appendice in una collezione di documenti.

`Help`

Indica un documento utilizzato come 'help' (informazioni aggiuntive, collegamenti ad altri documenti, etc.)

Vediamo di seguito alcuni esempi dell'elemento `link` con i suoi attributi.

Esempio 1:

Il documento corrente è in relazione con la pagina "**capitolo-14.html**" come movimento a documento successivo (rel='next'), conseguentemente la pagina "**capitolo-14.html**" (pagina correlata) è in relazione con il documento corrente come movimento al documento precedente (rev='prev').

```
<head>
  ...
  <link href="capitolo-14.html" rel="next" rev="prev" />
  ...
</head>
```

Esempio 2:

Il documento corrente utilizza come foglio di stile per la presentazione del suo contenuto il file `"styles/classic.css"` che è di tipo `"text/css"`.

```
<head>
  ...
  <link href="styles/classic.css" rel="stylesheet" type="text/css" />
  ...
</head>
```

5.1.3.4 Codice di scripting

`<script> … </script>`

L'elemento `script` consente di inserire codice di scripting all'interno del documento. Il Browser elabora il contenuto dell'elemento come comandi e dati dello Script[25].

Attributi:

- `language`, dichiara il linguaggio di scripting utilizzato;
- `type`, dichiara il tipo del contenuto dello script in formato MIME;
- `src`, URI del file contente il codice di scripting da eseguire, utilizzato nel caso in cui si voglia separare il codice dal documento HTML.

Vediamo ora alcuni esempi.

Esempio 1:

```
<script language="JavaScript" type="application/x-javascript">
  <!--
     Codice e dati JavaScript
  // -->
</script>
```

[25] L'elemento `script` può essere inserito anche all'interno dell'elemento `body`

In questo esempio il codice da eseguire è scritto nel linguaggio Javascript all'interno dell'elemento script.

Da notare che, per impedire che alcuni Browser, che non supportano l'elemento **script**, interpretino il codice di scripting come html, è necessario racchiudere il codice tra commenti: il codice di scripting è racchiuso tra commenti HTML (vedi paragrafo '*I commenti*') per non essere visualizzato dal Browser e la chiusura del commento HTML è commentata per il codice di scripting per non essere eseguita.

Esempio 2:

```
<script language="JavaScript" type="application/x-javascript"
        src="http://www.nic.it/script.js">
</script>
```

In questo esempio, il codice JavaScript è separato dal file HTML, il Browser carica il file **'script.js'** dal server contenente il codice JavaScript da eseguire.

5.1.3.5 Definizione degli stili

<style> … </style>

Il testo contenuto negli elementi **<style>** è considerato dal client parte delle regole di stile da applicare nella presentazione del documento.

Attributi:

- **type**, descrive il tipo di foglio di stile utilizzato in formato MIME, Esempio: **text/css** (CSS);
- **media**, indica la tipologia di Browser che interpreterà il documento, Esempio: **screen** (periferiche video), **braille** (dispositivi tattile), **aural** (periferiche audio), ecc;

Esempio:

```
<head>
  ...
  <style type="text/css" media="aural">
    <!--
     definizione dello stile
    // -->
  </style>
  ...
</head>
```

5.1.4 Corpo del documento

<body> … </body>

L'elemento **body** contiene il corpo del documento.

Il corpo del documento può contenere, sia una semplice sequenza di paragrafi testuali, che una struttura complessa costituita da testo formattato con immagini, tabelle, liste, ecc..

Esempio:

```
...                          ←  Definizione del tipo di documento

<html>
  <head> ... </head>         ←  Intestazione del documento

  <body> ... </body>         ←  Corpo del documento
</html>
```

Per l'elemento **body** non ci sono attributi specifici, essi sono stati tutti deprecati in favore dell'utilizzo dei CSS.

Come sottolineato spesso in questo documento, una presentazione di successo deve avere il testo ben organizzato in modo da creare un documento di impatto visivo. Organizzare un documento è il punto di forza dell'HTML il quale mette a disposizione una varietà di elementi che consentono un facile accesso alle informazioni da parte dell'utente.

Durante la progettazione di un documento è da tenere presente che gli elementi di markup, specialmente quelli per la formattazione del testo, non determinano l'esatto aspetto finale del documento, ma ne danno un'indicazione. La presentazione del documento varia, infatti, a seconda dell'interpretazione che il Browser da ai vari elementi.

Gli elementi messi a disposizione dall'HTML, quindi, consentono di variare il **significato** e, di conseguenza, la **presentazione** di porzioni di testo contenute in un documento. Un elemento informa il Browser che il testo racchiuso ha un preciso significato contestuale ed il Browser presenta il testo in modo consistente al suo significato.

Gli elementi dell'HTML possono essere suddivisi in quattro tipologie diverse:

1. Elementi di tipo '**Block level**', che racchiudono un intero blocco di testo e, tipicamente, generano un'interruzione di linea prima e dopo il testo in essi contenuto;
2. Elementi di tipo '**inline**', che racchiudono una porzione di testo senza interruzioni di riga;
3. Liste;
4. Tabelle;

Nei paragrafi successivi prenderemo in esame gli elementi principali di queste categorie.

5.1.4.1 Cenni sui frame

`<frameset>, <frame>, <noframes>`

Come già accennato nel paragrafo 5.1 Struttura del documento HTML, in questo documento non sarà approfondito l'uso dei frame poiché considerato altamente sconsigliato; in questo paragrafo vedremo le ragioni.

I frame permettono di dividere lo schermo in righe e colonne ognuna contenente un documento diverso. Per realizzare i frame è necessario utilizzare due diversi tipi di elementi: `frameset` e `frame`

- `frameset`, che è il contenitore degli elementi; contiene i frame che realizzano la suddivisione dello schermo e definisce il numero di righe e colonne nonché la loro dimensione assoluto o relativa;
- `frame`, che definisce quale documento (HTML o altro) è inizialmente inserito nel frame;
- `noframes`, che contiene il testo alternativo che è visualizzato dai Browser che non supportano i frame.

Anche se apparentemente utili, il loro utilizzo è sconsigliato per i seguenti principali motovi:

1. i frame non sono supportati da tutti browser e, probabilmente, non saranno più supportati nell'XHTML 1.1;
2. nelle pagine contenenti i frame, si nota spesso il malfunzionamento del tasto Back dei Browser che non riescono a capire qual'è la pagina dove devono tornare o qual è il frame;
3. si presentano problemi nella stampa della pagina; i Browser stampano il documento contenuto nel frame dove si trova il mouse in quel momento;
4. i motori di ricerca trovano il singolo documento contenuto nel frame e quindi, selezionando l'indirizzo trovato viene visualizzata la singola pagina e conseguentemente viene persa la struttura del sito.

Vedremo nei paragrafi successivi la metodologia corretta per realizzare le stesse funzionalità dei frame utilizzando elementi di struttura non deprecati come i *div* e personalizzandoli attraverso l'utilizzo dei CSS.

5.1.4.2 Elementi "block-level"

Tra gli Elementi 'Block level' vedremo nei prossimi paragrafi:

- I commenti
- Sezioni e paragrafi: `div`, `p`, `br`
- Testo preformattato: `pre`
- Intestazioni: `h1`, `h2`, ..., `h6`
- Linea orizzontale: `hr`

I commenti

`<!-- …-->`

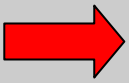
Prima di prendere in esame i principali elementi contenuti nel corpo del documento, vediamo cosa sono i commenti in HTML.

Un commento è un blocco di testo, all'interno del documento che non deve essere visto dall'utente. Tipicamente un commento viene utilizzato per dare informazioni sul documento ad altri autori che ne vedranno il codice sorgente.

Un commento inizia con la stringa `'<!--'` e finisce con la stringa `'-->'`; il blocco di testo compreso tra le due stringhe è ignorato dai Browser[26].

Esempio:

```
Linea1
<!-- linea di commento -->
Linea2
```



```
Linea1 Linea2
```

Sezioni e Paragrafi

In assenza di elementi di formattazione, il Browser distribuisce il testo spezzandolo a seconda della dimensione orizzontale dello spazio visibile.

Per controllare la disposizione del testo l'HTML utilizza i seguenti elementi di blocco:

- `<div> ... </div>`, che definisce una sezione del documento;

[26] ATTENZIONE: teoricamente non è consentito inserire un terzo carattere '-' dopo l'apertura del commento anche se molti Browser lo consentono

- `<p> ... </p>`, che delimita un paragrafo;
- `<br />`, che forza un ritorno a capo della linea.

La disposizione del testo è caratterizzabile completamente tramite i CSS, che vedremo più in là nel documento.

L'elemento div

`<div> … </div>`

L'elemento `div` divide il documento in sezioni separate e distinte e può essere utilizzato solo per scopi organizzativi. Esso è cioè un contenitore generico di stile utilizzato per la formattazione dei blocchi che diventa un strumento potente se utilizzato con i CSS.

Questo elemento da la possibilità di etichettare una sezione in modo che possa essere referenziata da un collegamento ipertestuale, identificata e gestita da processi automatici o utilizzata per definire uno stile di presentazione particolare.

Attributi:

- `id="string"`, il valore dell'attributo è utilizzato come etichetta identificativa della sezione;
- `class="string"`, il valore dell'attributo corrisponde al nome di uno stile definito altrove.

Da notare che gli attributi `id` e `class` sono attributi comuni e quindi utilizzabili su tutti gli elementi del body.

Gli elementi p e br

`<p> … </p>`

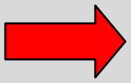
L'elemento `p` definisce l'inizio e la fine di un paragrafo.

Prima dell'inizio di un nuovo paragrafo, il browser inserisce una riga bianca. All'interno di un paragrafo, il Browser organizza ogni sequenza di caratteri e/o immagini contenuti nel paragrafo in un testo formattato secondo la dimensione dedicatagli.

Se si vuole forzare una linea a capo si deve inserire l'elemento `br`.

Esempio:

```
<p>
  Primo paragrafo
</p>
<p>
  Linea1: secondo
  paragrafo <br /> Linea2:
  secondo paragrafo
</p>
```



Primo paragrafo

Linea1: secondo paragrafo
Linea2: secondo paragrafo

Testo preformattato

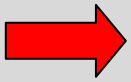
`<pre> … </pre>`

L'elemento **`pre`** definisce l'inizio e la fine di una porzione di testo che il Browser deve presentare esattamente come formattato nel codice sorgente mantenendo anche gli spazi bianchi.

Il Browser presenta il testo contenuto nell'elemento **`pre`** con un carattere 'monospace'. L'elemento **`pre`** non può contenere elementi di tipo *block* (intestazioni, paragrafi, ecc.).

Esempio:

```
<pre>
   Chi non ammette
   l'insondabile mistero
   non può essere neanche
   uno scienziato.
   (Albert Einstein)
</pre>
```



```
Chi non ammette
l'insondabile mistero
non può essere neanche
uno scienziato.
(Albert Einstein)
```

Intestazioni

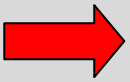
`<hn> … </hn> n=1, .. 6`

Le intestazioni sono utilizzate per definire i titoli ed i sottotitoli di diverse sezioni all'interno del documento. In HTML esistono sei livelli di intestazioni; tipicamente **h1** è utilizzato per definire il titolo del documento, **h2** per i titoli delle sezioni e così via per le sotto-sezioni.

Il testo contenuto negli elementi è presentato a seconda della tecnologia messa a disposizione dal client. Tipicamente un Browser testuale visualizza il testo in grassetto e diminuisce la dimensione del carattere con l'aumentare del parametro 'n'.

Esempio:

```
<h1> Titolo </h1>
<h2> Paragrafo 1</h2>
<h3> Paragrafo 1.1 </h3>
<h4> Paragrafo 1.1.1 </h4>
```



Titolo

Paragrafo 1

Paragrafo 1.1

Paragrafo 1.1.1

Da sottolineare che le intestazioni devono essere utilizzate per strutturare il documento (titoli, sottotitoli, ecc.) e non per la formattazione. Spesso le intestazioni sono utilizzate impropriamente per cambiare il font di una sezione o di un intero documento.

Linea orizzontale

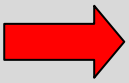
`<hr />`

La linea orizzontale è utilizzata per separare visivamente sezioni diverse all'interno di un documento.

L'elemento **hr** dice al Browser di disegnare una linea orizzontale; le caratteristiche grafiche della linea dipendono dal Browser ma solitamente la linea si estende per tutta la dimensione orizzontale del blocco che la contiene, forza a capo la linea successiva, riporta l'allineamento del paragrafo all'impostazione di default (allineamento a sinistra) e non crea spazi aggiuntivi sopra e sotto la linea disegnata. La presentazione grafica della linea orizzontale è, comunque, personalizzabile con l'utilizzo dei CSS.

Esempio:

```
Linea 1
<hr />
Linea 2
```



Linea 1

Linea 2

5.1.4.3 Elementi "inline"

Gli elementi 'in line' che tratteremo in questo paragrafo sono:

- Testo enfatizzato: `em`, `strong`
- Testo da identificare: `span`
- Altri elementi inline: `address`, `abbr`, `acronym`, `code`, `cite`

Testo enfatizzato

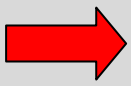
`<em> … </em>`

L'elemento `em` indica al Client di presentare il testo in esso contenuto con particolare enfasi. Questo elemento viene utilizzato per presentare termini nuovi o con uno specifico significato o concetto.

Tipicamente il testo è presentato in italico ma particolari Browser potrebbero realizzare l'enfasi con determinati effetti speciali.

Esempio:

```
testo <em>enfatizzato</em>
```



testo *enfatizzato*

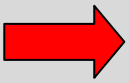
Testo molto enfatizzato

`<strong> … </strong>`

L'elemento `strong`, come `em`, indica al Client di presentare il testo contenuto con enfasi. L'interpretazione del client dell'elemento `strong` è, però, più forte di quella adottata per l'elemento `em`; tipicamente il testo viene presentato in grassetto

Esempio:

```
testo
<strong>
molto enfatizzato
</strong>
```



testo **molto enfatizzato**

Testo da identificare

`<span> … </span>`

L'elemento `span` viene utilizzato quando si vuole identificare una porzione di testo da personalizzare successivamente tramite i CSS.

Di default non esiste alcuna formattazione associata allo `span` ma, con i CSS, il testo contenuto può essere oltre che formattato anche ricercato in futuro.

L'elemento `span` è il corrispondente dell'elemento `div`, la differenza tra i due è che il primo viene utilizzato all'interno di una riga, mentre il secondo è un elemento di blocco.

Altri elementi 'inline'

Di seguito sono indicati altri elementi di tipo 'in line' comunemente utilizzati ed alcuni esempi.

`address`:

Contiene i contatti di riferimento per il documento, come per esempio l'autore; questo elemento è solitamente inserito all'inizio o alla fine del documento

`abbr`:

Contiene una forma abbreviata (Es: Sig., Prof., ecc.) .

`acronym`:

Indica un acronimo (Es: FBI, NATO, ecc.) .

`code`:

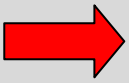
Contiene un frammento di codice sorgente scritto in un qualsiasi linguaggio di programmazione.

`cite`:

Contiene una citazione o un riferimento preso da un altro documento.

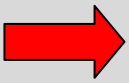
Esempi:

```
<address>
  Paolino Paperino <br />
  Via del deposito, 13 <br />
  Paperopoli <br />
</address>
```



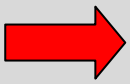
Paolino Paperino
Via del deposito, 13
Paperopoli

```
Per informazioni contattare <br />
il <abbr>Sig.<abbr/> Rossi
```



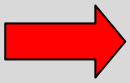
Per informazioni contattare il Sig. Rossi

```
Il <acronym title="World Wide Web">
WWW </acronym> <br /> è dove si
pubblicano le pagine XHTML
```



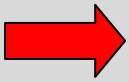
Il WWW è dove si pubblicano le pagine XHTML

```
<code>
  if (i > 0) i--;
</code>
```



```
if (i > 0) i--;
```

```
Come <cite>O. Wilde</cite>
scrisse <br /> una volta ...
```



Come *O. Wilde* scrisse una volta ...

5.1.4.4 Liste

L'HTML permette di organizzare le informazioni contenute in un documento in liste formattate.

In HTML esistono tre tipi di liste:

- Lista non ordinata
- Lista ordinata
- Lista di definizione

Ogni lista è definita da due tipi di elementi:

1. L'elemento che apre e chiude la lista, cioè il contenitore della lista stessa;
2. Gli elementi che racchiudono i componenti della lista.

Lista non ordinata

`<ul> .. </ul>`

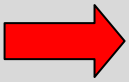
Una lista non ordinata è utilizzata per elencare un insieme di elementi che non hanno una sequenza o un ordine particolare (Es: la lista della spesa).

- L'elemento **`<ul>…</ul>`** indica al Browser che il testo contenuto è una lista non ordinata
- L'elemento **`<li>…</li>`** identifica un componente all'interno della lista

Il Browser, tipicamente, inserisce un cerchio pieno all'inizio di ogni componente che viene disposto su una nuova linea e indentato rispetto al margine sinistro del documento

Esempio:

```
<ul>
  <li> acqua </li>
  <li> vino </li>
</ul>
```



- acqua
- vino

Lista ordinata

`<ol> .. </ol>`

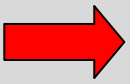
Una lista ordinata viene utilizzata per elencare un insieme di elementi il cui ordine è importante (Es: una sequenza di istruzioni, l'indice di un documento)

- L'elemento **`<ol>…</ol>`** indica al Browser che il testo contenuto è una lista ordinata
- L'elemento **`<li>…</li>`** identifica un componente all'interno della lista

Il Browser, tipicamente, inserisce automaticamente un numero sequenziale all'inizio di ogni componente il quale viene formattato come una lista non ordinata.

Esempio:

```
 <ol>
  <li> Introduzione </li>
  <li> Capitolo </li>
</ol>
```



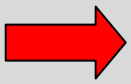
1. Introduzione
2. Capitolo

Liste annidate

Le liste ordinate e non ordinate possono essere annidate tra loro; vediamo alcuni esempi.

Esempio1: Liste non ordinate annidate

```
<ul>
<li> frutta
    <ul>
      <li> mele </li>
      <li> pere </li>
    </ul>
</li>
<li> pasta </li>
</ul>
```

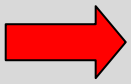


- frutta
  - mele
  - pere
- pasta

Per le liste non ordinate il Browser, solitamente cambia la tipologia del cerchio a seconda del livello di indentazione.

Esempio2: Liste ordinate annidate

```
 <ol>
  <li> introduzione </li>
  <li> capitolo
    <ol>
      <li>sottocapitolo</li>
      <li>sottocapitolo</li>
    </ol>
  </li>
</ol>
```



1 Introduzione

2 capitolo

2.1 sottocapitolo

2.2 sottocapitolo

Per le liste ordinate i Browser, di default, iniziano la numerazione dei componenti della lista da 1, indipendentemente dal livello di indentazione; per una migliore leggibilità del documento, è a volte necessario cambiare il tipo di numerazione utilizzando i CSS.

Naturalmente, possono essere annidate contemporaneamente sia liste ordinate, che non ordinate.

Lista di definizione

`<dl>…</dl>`

Una lista di definizione è utilizzata tipicamente per presentare un glossario o una lista di termini

Una lista di definizione è contenuta tra i tag `<dl>…</dl>`

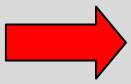
Ogni componente della lista è composto da due parti:

- Un termine, delimitato dai tag `<dt>` e `</dt>`
- La sua definizione o spiegazione, delimitata dai tag `<dd>` e `</dd>`

Di default, il Browser visualizza il termine allineato al bordo sinistro e la sua definizione a capo e indentata.

Esempio:

```
<dl>
<dt>IIT</dt>
  <dd>Istituto di Informatica e
     Telematica
  </dd>
<dt> ISTI </dt>
  <dd>Istituto di Scienza e Tecnologie
     dell'Informazione
  </dd>
</dl>
```



IIT

Istituto di Informatica e Telematica

ISTI

Istituto di Scienza e Tecnologia dell'Informazione

CSS: parte I

6.1 Cascading Style Sheet (CSS)

L'HTML permette di creare una struttura della pagina e di fornire una formattazione al testo ma con il solo HTML non si possono controllare tutti gli aspetti di come il testo sarà presentato.

I CSS (Cascading Style Sheet) nascono invece con il preciso scopo di specificare come il testo verrà presentato e lavorano insieme all'HTML per descrivere la *struttura del documento* (con l'HTML) e la sua *presentazione (con i CSS)*.

Nonostante gli elementi e gli attributi delle pagine HTML **dovrebbero** servire solo per definire la struttura dei documenti (collegamenti, paragrafi, titoli, sezioni,...), l'idea iniziale è stata parzialmente distorta e agli elementi HTML sono stati aggiunti molti attributi che indicano proprietà relative alla presentazione; sono nati per esempio attributi ed elementi per cambiare:

- il colore del testo;
- l'allineamento;
- la famiglia e la dimensione dei caratteri
- ...

Agli attributi e agli elementi di formattazione si deve preferire l'uso dei Cascading Style Sheet (CSS), che permettendo la separazione tra contenuto e presentazione facilitano lo sviluppo dei siti web ed il loro mantenimento.

Come per l'HTML ci sono due modi per scrivere gli stylesheet:

- "a mano"
- tramite un software dedicato o un software di Web Design (dreamweaver, homesite, etc.)

A volte i web designer scrivono i CSS con strumenti automatici e poi li ritoccano a mano successivamente.

Noi scriveremo "a mano" gli stylesheet per capirne bene la sintassi, tenendo presente che:

- "a mano" è più difficile e laborioso ma permette di mantenere il controllo "della situazione"
- tramite un software di Web Design che abbia questa funzionalità inizialmente lo sviluppo è più rapido ma difficilmente ci permette di comprendere la sintassi dei CSS

6.2 Composizione degli stylesheet

Uno stylesheet (foglio di stile) è un insieme di **regole** che definisce la presentazione di un documento, attraverso queste regole si specifica l'aspetto stilistico di uno o più elementi HTML.

Gli aspetti che possono essere specificati possono riguardare media diversi, per esempio è possibile specificare regole che valgono per la presentazione a video, per la stampa, per la riproduzione audio, etc.

Durante questa documentazione noi faremo riferimento solo alla presentazione a video degli elementi e tratteremo quindi solamente le proprietà che si riferiscono a questo media.

Definizioni

stylesheet	insieme di regole che definisce la presentazione di un documento
regola	istruzione che permette di specificare l'aspetto stilistico di uno o più elementi

Il primo esempio di regola:

```
h1 { color: green }
```

Le regole utilizzano un linguaggio semplice ed intuitivo che rende semplice capirne la semantica, il linguaggio naturale di riferimento è naturalmente l'inglese. Nell'esempio soprastante la regola specifica che il colore (**color**) del testo degli elementi di tipo **h1** venga impostato a *verde (green)*.

Una regola è composta di due parti:

- **selettore -** la parte prima delle parentesi graffe - specifica gli elementi assoggettati a questa regola ed è il collegamento tra il documento HTML e lo stile
- **dichiarazione** - la parte dentro le graffe - è la parte della regola che imposta il comportamento stilistico degli oggetti coinvolti

```
  h1    { color: green }
-----    --------------
  |            |
selettore  dichiarazione
```

Il selettore dell'esempio precedente è un selettore per **tipo** ovvero individua gli elementi assoggettati alla dichiarazione tramite il loro tipo ovvero “h1”; quindi tutti gli elementi di tipo h1 sono soggetti alla dichiarazione, vedremo nel seguito che esistono altri tipi di selettore.

Una dichiarazione è a sua volta composta di due parti distinte, la **proprietà** ed il **valore**, le quali sono separate da ":" (due punti):

- **proprietà** - la parte prima dei due punti - descrive una caratteristica che un oggetto può avere
- **valore** - la parte dopo i due punti - specifica quale comportamento deve avere la proprietà selezionata

```
h1 { color: green }
     -----  -----
       |      |
  proprietà  valore
```

La proprietà **color** specifica il colore del testo contenuto all'interno dell'elemento.

Tutte le parti di una regola:

```
  selettore
   |
   |  proprietà  valore
   |      |        |
   -    -----    -----
  h1  { color : green }
        -------------
              |
         dichiarazione
```

Altri esempi di regole:

```
h2 { font-weight: bold} – imposta il carattere in grassetto
h3 { text-align: center } - il testo viene centrato all'interno
dell'elemento
h4 { color: red } - il colore del testo è rosso
```

Nota:

*I possibili valori della proprietà **color** possono essere espressi sia con una notazione mnemonica (white, black, red) che con altri tipi di notazione che vedremo in dettaglio più avanti (RGB ed esadecimale), nell'esempio è riportata la notazione mnemonica del colore verde (il valore **green**).*

6.3 Raggruppare le regole

Scrivendo le regole CSS ci si rende subito conto che scrivere una regola per ogni tipo di elemento può essere un compito molto lungo anche se supportati da un editore specifico per scrivere i CSS. Esistono delle facilitazioni che permettono di raggruppare le regole abbreviando in questo modo gli stylesheet ed ottenendo i seguenti vantaggi:

- riduzione delle dimensioni del CSS;
- semplificazione della gestione manuale;
- diminuzione del tempo di trasferimento dalla rete.

I CSS includono vari meccanismi per scrivere stylesheet brevi.

6.3.1 Raggruppare per dichiarazione

È possibile raggruppare le regole che hanno dichiarazioni uguali, in una unica regola mettendo:

- come selettore, la lista dei selettori separati da virgole
- come dichiarazione, la dichiarazione comune a tutte le regole

Le tre regole:

```
h1 { font-weight: bold }
h2 { font-weight: bold }
h3 { font-weight: bold }
```

Possono essere rappresentate con la regola:

```
h1, h2, h3 { font-weight: bold }
```

Nota:

La proprietà "font-weight" imposta il peso del font interessato.

I valori possibili sono:

1. *normal*
2. *bold*
3. *bolder*
4. *lighter*
5. *100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900*

6.3.2 Raggruppare per selettore

Più regole, associate allo stesso selettore, possono essere contratte tramite un'unica regola in cui:

- il selettore è lo stesso
- la dichiarazione è la lista delle dichiarazioni delle varie regole separate dal carattere ";".

Le tre regole:

```
h1 { color: green }
h1 { font-weight: bold }
h1 { text-align: center }
```

Possono essere raggruppate nell'unica regola:

```
h1 { color: green;
     font-weight: bold;
     text-align: center;
   }
```

Il carattere ";" dopo l'ultimo elemento della lista è facoltativo.

6.3.3 Raggruppare per dichiarazione e per selettore

È possibile raggruppare sia per dichiarazione che per selettore utilizzando le regole prima esposte, è infatti possibile scrivere:

```
h1, h2, h3 { color: green;
             font-weight: bold;
             text-align: center;
           }
```

Questo assegna le tre dichiarazioni specificate agli elementi h1, h2 e h3.

6.4 Le specifiche CSS

I CSS (Cascading Style Sheet) sono formalmente descritti in due raccomandazioni ed un Working Draft (almeno nel momento in cui questo testo è stato scritto) del W3C:

- CSS 1 level → 1 http://www.w3.org/TR/1999/REC-CSS1-19990111
- CSS 2 → http://www.w3.org/TR/REC-CSS2
- CSS 2 level 1 (CSS 2.1) → http://www.w3.org/TR/CSS21

CSS1, pubblicata nel 1996 e rivista nel 1999, descrive attraverso un semplice modello alcune proprietà che si riferiscono principalmente alle presentazioni a video. CSS 1 descrive circa 50 proprietà tra cui, per esempio, `color` e `font-weight`.

CSS2, pubblicata nel 1998, contiene tutte le proprietà di CSS 1 e ne aggiunge 70; principalmente proprietà che descrivono la rappresentazione audio (aural), quella per le stampe (print) e l'interruzione delle pagine (page breaking).

CSS 2.1, di cui l'attuale working Draft è del 13 giugno 2005, è costruita sopra la specifica CSS2 di cui corregge alcuni errori e propone caratteristiche prese da pratiche di utilizzo dei CSS generalmente accettate.

In seguito non faremo distinzione tra proprietà CSS 1 e/o CSS 2 e/o CSS 2.1 a meno che sia per qualche motivo rilevante.

6.4.1 CSS 2.1 vs CSS 2

La comunità degli sviluppatore ed utilizzatore di CSS ha fatto molta esperienza con la specifica CSS dalla data della sua pubblicazione nel 1998 fino ad oggi. Gli errori che sono stati riscontrati nella specifica sono stati corretti di volta in volta con una serie di “Errata”, ma visto la quantità di modifiche il gruppo di lavoro del W3C sui CSS ha deciso di pubblicare una versione CSS 2.1 in attesa dell'arrivo della nuova specifica CSS3.

La specifica CSS 2.1 viene incontro ad una serie di situazioni:

- Maintiene la compatibilità con le porzioni di CSS 2 che sono state largamente accettate ed implementate ed incorpora tutti gli “Errata” di CSS 2 pubblicati;
- Quando le implementazioni più diffuse differiscono dalla specifica CSS 2, modifica la specifica in accordo con le pratiche generalmente accettate;
- Rimuove le caratteristiche di CSS che non sono state implementate e sono state respinte dalla comunità CSS;

- Rimuove le caratteristiche di CSS 2 che sono state rese obsolete da CSS 3, incoraggiando al loro posto l'utilizzo delle proposte di CSS 3;
- Aggiunge (pochi) nuovi valori per alcune proprietà, di cui si è sentita la necessità dopo la pubblicazione di CSS 2.

Purtroppo alcuni stylesheet scritti con CSS 2 non sono compatibili con la specifica CSS2.1, ma d'altra parte l'adozione delle specifiche CSS 2.1 garantisce una maggiore garanzia di trovare browser (user agent più in generale) in grado di interpretare correttamente lo stylesheet ed inoltre garantisce di preservare la compatibilità con il futuro.

6.4.2 CSS 3: il futuro

Il futuro dei CSS sono le specifiche CSS 3 attualmente in fase di "Working Draft" che vedranno la suddivisione della specifica CSS in una serie di moduli, ognuno per un aspetto particolare dei CSS, per esempio: CSS3 module: Syntax, CSS3 module: the box model, CSS3 Text Module, ...

6.5 Associare gli stylesheet all'HTML

Esistono tre modi per associare gli stylesheet ad un documento HTML:

- applicando un attributo `style` ad un elemento del documento
- applicando l'elemento `style` al documento HTML
- collegando uno stylesheet esterno tramite l'elemento `link`

Il primo metodo - l'attributo `style` - prevede che lo stile sia scritto all'interno del file HTML; il secondo metodo - l'elemento `style` - può far si che lo stile sia scritto internamente o in file esterno; il terzo metodo - l'elemento `link` - utilizza un file esterno per lo stylesheet.

6.5.1 L'attributo style

L'attributo `style` permette di specificare regole CSS per l'elemento corrente.

È consigliato solo per elementi che hanno un comportamento stilistisco "singolare", cioè non comune ad altri elementi.

Esempio

```
<span style="font-weight: bold">
 Questo testo è in grassetto
</span>
```

Questo testo è in grassetto

L'attributo `style` viene utilizzato per non scrivere troppe informazioni, utilizzate per singoli elementi, nel CSS e velocizzare la scrittura del codice.

6.5.2 L'elemento style

Associare lo stile con l'elemento `style` :

```
...
<head>
  ...
  <style type="text/css">
    <!--
      h1 { color: green }
    -->
  </style>
</head>
<body>
  <h1>Alcuni buoni motivi per utilizzare
     i fogli di stile
  </h1>
  <p>Scrivere pagine HTML con stylesheet
   associati, dividendo così lo stile dalla
   presentazione, ha diversi vantaggi:</p>
  <ul>
    <li>Lo stile delle pagine è più
     omogeneo</li>
    <li>Si può cambiare lo stile senza
cambiare
   la struttura delle pagine</li>
    <li>...</li>
...
```

Alcuni buoni motivi per utilizzare i fogli di stile

Scrivere pagine HTML con stylesheet associati, dividendo così lo stile dalla presentazione, ha diversi vantaggi:

- Lo stile delle pagine è più omogeneo
- Si può cambiare lo stile senza cambiare
- la struttura delle pagine
- ...

Nota 1: Specificare il `type` dello style

Per adesso nel HTML esiste un solo tipo di stylesheet, indicato dal `type="text/css"`, ma in futuro potrebbero esisterne altri, specificare quale tipo di stile utilizzare è buona norma

Nota 2: Mettete lo stylesheet dentro un commento HTML.

Dentro l'elemento style lo stylesheet è racchiuso all'interno di un commento HTML, come mai?

I browser vecchi (quelli molto vecchi, tipo Explorer 2 o Netscape 1) non comprendevano l'elemento `<style>`. Inserendo lo stylesheet dentro un commento, i browser non facevano vedere il contenuto dell'elemento a video.

I browser "nuovi" ignorano il commento.

6.5.3 L'elemento link

Un esempio di utilizzo dell'elemento `link`:

```
...
    <head>
      <title>Perché usare i fogli di stile</title>
      <link href="base.css" type="text/css" rel="stylesheet" />
      ...
    </head>

    <body>
      <h1>Alcuni buoni motivi per utilizzare i fogli di
stile</h1>
      ...
```

Dove l'attributo `href="base.css"` indica il file in cui è dichiarato il CSS.

L'attributo `rel="stylesheet"` dell'elemento `link` indica al browser che l'oggetto che stiamo collegando deve essere trattato come uno stylesheet del documento HTML corrente.

Il risultato è uguale ad utilizzare l'elemento `style`.

6.6 Gli Stylesheet esterni

Nella maggior parte dei casi è preferibile **utilizzare uno stylesheet esterno** rispetto alle altre soluzioni, per le seguenti motivazioni:

- **Riusabilità** - Inserendo le informazioni sullo stile all'interno di un unico file è possibile collegarlo a più documenti HTML. In questo modo posso definire uno o più fogli di stile che possono essere utilizzati in uno o più siti permettendo una presentazione uniforme e coerente in tutte le pagine senza bisogno di riscrivere lo stilo completamente per ogni pagina.
- **Prestazioni** - Il browser può caricare lo stylesheet una sola volta e poi tenerlo in cache. Non inserendo lo stile nelle pagine HTML si evita di ridefinire nuovamente il codice diminuendo la quantità totale di file scaricati (file HTML+ file CSS);
- **Possibile scelta dell'utente** - E' possibile indicare più stylesheet associati ad un documento e dare la possibilità all'utente di selezionare quale utilizzare. In questo modo non solo si può cambiare completamente la presentazione tramite fogli di stile diversi ma si permette anche la possibile differenziazione della presentazione in funzione del device (video, stampa, presentazione, etc.)

Durante le esercitazioni del corso utilizzeremo sempre stylesheet esterni.

6.7 Struttura ad albero ed ereditarietà

I CSS sfruttano una delle caratteristiche fondamentali del HTML, la sua **struttura ad albero**.

La struttura ad albero permette di utilizzare i concetti di padre e figlio/i per gli elementi HTML, per cui come si vede dall'esempio seguente, gli elementi `h1` e `p` sono entrambi figli di `body`

```
...
  <body>
    <h1>Ereditarietà delle proprietà</h1>
    <p>Gli elementi ereditano le proprietà dello stile dal
       padre
    </p>
  </body>  ...
```

I CSS sfruttano questa caratteristica facendo si che gli elementi HTML ereditino le proprietà dello stile dal padre.

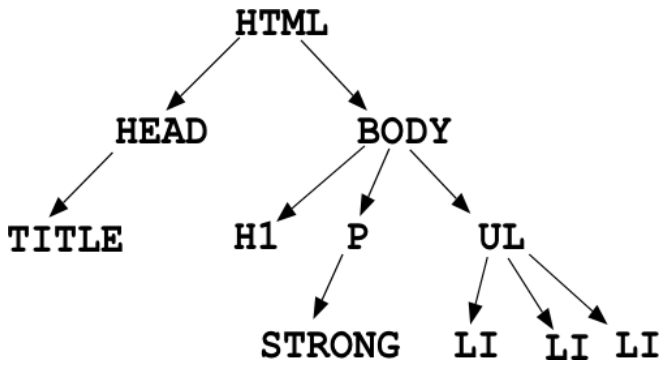


Figura 15 - Esempio di Struttura ad albero

impostando una regola sull'elemento `body`, per esempio:

```
body { color: green }
```

la regola viene applicata a tutti i suoi figli di body, quindi **sia il contenuto di `h1` che di `p`, che di `ul` sarà di colore** *verde*.

6.7.1 Sovrascrivere l'ereditarietà

Non sempre però si vuole che i figli ereditino le proprietà dal padre.

Per esempio:

```
body { color: green }
h1   { color: blue }
```

- la prima regola imposta il colore dell'elemento body (e quindi di tutti i suoi figli, compreso `h1`) a verde
- la seconda regola imposta il colore dell'elemento `h1` a blu.

Queste due regole quindi hanno un conflitto di impostazione sul colore dell'elemento `h1`, in questo caso la regola che "vince" è: `h1 { color: blue }`

questo perché si riferisce in modo *più specifico* all'elemento `h1` rispetto all'altra.

Nel caso sopra dichiarare una regola sull'elemento `h1` è più rilevante che dichiararla su un elemento che contiene `h1` (`body`)

I CSS hanno un meccanismo interno di risoluzione dei conflitti sulle regole, l'idea generale è comunque che più è "specifico" il selettore delle regola più è alta la probabilità che vinca eventuali conflitti.

6.7.2 Proprietà che non si ereditano

Come regola generale gli elementi figli ereditano le proprietà dai padri come descritto precedentemente.

Alcune proprietà però non seguono questa regola e ci sono dei buoni motivi per non farlo.

La proprietà **background** è un esempio di proprietà che non si eredita dal padre.

Una lista delle proprietà CSS con le relative caratteristiche (compreso se sono ereditate o meno dal padre) è reperibile all'url

- http://www.w3.org/TR/REC-CSS2/propidx.html

6.7.3 Proprietà che non si ereditano: background

Prendiamo un esempio:

```
...<body>
     <h1>Proprietà che non si ereditano</h1>
     <p>Altre proprietà che non si ereditano sono:</p>
     <ul>
       <li>border</li>
       <li>margin</li>
       ...
     </ul>...
```

con il seguente stylesheet associato:

```
body {
  background: url(sfondo.png);
  color: black;
}
```

Figura 16 - Esempio di proprietà background

La proprietà "background", osservazione:

L'immagine di sfondo copre tutto lo sfondo della pagina, anche lo sfondo degli elementi `h1` e `p`

Nota:

- la copertura di `h1` e `p` è dovuta allo sfondo trasparente dei due elementi (quello predefinito) e non all'ereditarietà!

Ci sono due ragioni per cui la proprietà non si eredita:

1. l'immagine di sfondo è allineata rispetto all'elemento che la contiene
2. gli sfondi trasparenti sono più veloci da visualizzare rispetto a sfondi con la stessa immagine

L'effetto se la proprietà "background" fosse ereditata sarebbe questo:

Figura 17 - Effetto della proprietà background non acquisita

Come di vede dall'immagine lo sfondo “ricomincerebbe” ogni qual volta si incontra un nuovo elemento HTML, portando ad un effetto indesiderato di box ognuno con il proprio sfondo.

6.8 Un utilizzo comune dei CSS: i font

Una delle proprietà più frequenti da impostare è il font che permette di specificare come saranno presentati i caratteri che vogliamo utilizzare.

Quelli che seguono sono esempi di utilizzo di font diversi.

Questo è il carattere arial

Serif in grassetto

Testo in fantasy

Oltre il tipo di font da utilizzare è inoltre possibile specificare diverse proprietà tra cui: la dimensione, il peso, la famiglia, etc..

6.8.1 font: "font-size"

La proprietà font-size imposta la dimensione del font. La tabella successiva riassume le sue caratteristiche fondamentali della proprietà.

Nome:	font-size
Valori:	<altezza> | <percentuale> | <dimensione-relativa> | <dimensione-assoluta>
Valore Iniziale:	medium
Si applica a:	tutti gli elementi
Ereditata:	si
Percentuali:	relativa alla dimensione del font del padre

6.8.2 Font-size specificato per altezza o in percentuale

La dimensione del font si può esprimere sia specificando l'altezza del font sia in percentuale rispetto al padre dell'elemento.

L'altezza del font può essere espressa in:

- unità assolute: millimetri (mm), centimetri (cm), pollici (in), punti (pt), pica (pc)
- unità relative: em, ex
- pixel

Queste unità di misura sono valide anche per tutte le altre proprietà che hanno bisogno di specificare delle dimensioni.

6.8.3 Unità di misura

Unità di misura	Descrizione
mm	Misurata in millimetri.
cm	Misurata in centimetri.
in	Misurata in pollici (inch). Un pollice è 25.4 mm
pt	Misurata in punti (point). Un punto è 1/72 di pollice.
pc	Misurata in pica. Un pica sono 12 punti (point).
px	Misurata in pixel.
em	Misura relativa alla larghezza dal carattere "m" dell'elemento padre.
ex	Misura relativa all'altezza del carattere "x" dell'elemento padre.
%	Misurata in percentuale relativa all'elemento padre.

È preferibile utilizzare le unità relative, per garantire la scalabilità dei font in funzione dei dispositivi.

6.8.4 Unità di misura relative

Le grandezze `ex`, `em`, `%` sono unità di misura relative.

em

Una unità `em` è la misura della larghezza del font dell'elemento padre di quello corrente.

Se il font di un elemento `p` è `10 pixel`, inserendo un elemento `strong` con font-size di `2 em`, il font-size dentro lo strong sarà `due volte` quella del `p` cioè `20 pixel`.

ex

Una unità `ex` è la misura dell'altezza del font x minuscolo dell'elemento padre.

Se il font di un paragrafo è `10 pixel`*, inserendo uno* `strong` *con font-size di* `2 ex`*, il font-size dentro lo strong sarà* `due volte` *l'altezza del carattere x minuscolo del* `p`.

%

Il 100% corrisponde all'altezza del font dell'elemento padre.

Se il font di un elemento `p` *è* `10 pixel`*, inserendo un elemento* `strong` *con font-size di* `200%`*, il font-size dentro lo strong sarà* `due volte` *quella del* `p` *cioè* `20 pixel`.

6.8.5 Esempi di font-size per altezza

Esempi di font-size

Font size 12 pt	Font size 12pt
Font size 16 pt	Font size 16pt
Font size 1.5pc	Font size 1.5pc
Font size 1.5em	Font size 1.5em (questo invece è il padre)
Font size 200%	Font size 200%(questo invece è il padre)

6.8.6 Font-size per valore assoluto

1. È possibile specificare una lista di valori per la dimensione del font.
2. La tabella con le dimensioni reali è mantenuta dal browser.
3. Tra una dimensione di font e la successiva ci dovrebbe essere un fattore del 120% circa.

Possibili valori di font-size

xx-small	abcdefghilmnopqrstuvz
x-small	abcdefghilmnopqrstuvz
medium	abcdefghilmnopqrstuvz
large	abcdefghilmnopqrstuvz
x-large	abcdefghilmnopqrstuvz
xx-large	abcdefghilmnopqrstuvz

6.8.7 Font-size per valore relativo

Il font-size relativo specifica la dimensione del font in funzione di quella dell'elemento padre.

La scelta della dimensione precisa viene lasciata al browser.

medium

La stessa dimensione del padre

larger

Più grande del padre

smaller

Più piccolo del padre

Il valore medium è proprio il valore predefinito per la proprietà font-size, quindi se non si specifica nessun font-size nel CSS, il browser utilizza come dimensione quella che lui stesso (o l'utente) ha impostato come predefinita. Alcuni browser permettono anche di poter far si che le scelte nello stylesheet per i font non siano rispettate e di utilizzare dei font decisi autonomamente con dimensione a piacimento dell'utente (o quelle predefinite del browser).

6.8.8 Font: "font-weight"

La proprietà font-weight imposta il peso del font.

Nome:	font-weight
Valori:	normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900
Valore Iniziale:	normal
Si applica a:	tutti gli elementi
Ereditata:	si
Percentuali:	N/A

6.8.9 Esempi di font-weight

esempio di font-weight:100

esempio di font-weight:900

font-weight: normal corrisponde a *font-weight:400*

font-weight: bold corrisponde a *font-weight:700*

font-weight: bolder (*font-weight: smaller*) imposta il peso del carattere ad un valore maggiore (minore) rispetto a quello del padre. Quanto più grande (più piccolo) dipende dai font e dal browser.

Nonostante la possibilità fornita dalla specifica CSS di specificare il valore in questo modo la maggior parte dei browser ignorano i valori specifici ed utilizzano il font-weight normal per i valori fino al 400 ed il font-weight bold per quelli superiori a 400, come riportato nell'esempio.

6.8.10 Famiglie di font-family

I font sono suddivisi in famiglie:

- serif

  i font con grazie, ovvero “stondature” per aggraziare i caratteri, per esempio: Times Roman, Garamond, e Palatino

- sans-serif

  i font senza grazie, come Helvetica, Arial

- monospace

  i font che hanno caratteri di larghezza fissa come per esempio Courier

- *cursive*

  i font corsivi o italici

- fantasy

  i font di tipo decorativo come per esempio Comic

6.8.11 Font specifici

Esempi di font specifici sono:

- Verdana
- arial
- Garamond
- Bodoni
- Comics

6.8.12 Font: font-family

La proprietà font-family permette di specificare il tipo di font o la famiglia di font da utilizzare.

Nome:	font-family
Valori:	[<famiglia-specifica>,]* | [<famiglia-specifica> | <famiglia-generica>]
Valore Iniziale:	dipende dal browser
Si applica a:	tutti gli elementi eccetto quelli rimpiazzati (per esempio: immagini)
Ereditata:	si
Percentuali:	N/A

Poiché i font presenti sul computer dell'utente possono essere i più svariati e non è detto che siano presenti i font specificati nella proprietà font-family, i CSS mettono a disposizione di specificare molti font specifici ed opzionalmente un font generico. Il browser utilizzerà il primo font che ha a disposizione leggendo la lista dei font specifici da sinistra verso destra; nel caso non ne abbia nessuno utilizzerà uno di quelli che ha a disposizione prelevandolo dalla lista dei font disponibili per la famiglia specificata (se si è specificata). Se non si specifica nessuna famiglia di font, se il browser non possiede nessuno dei font specifici dichiarati il browser può decidere arbitrariamente quale font utilizzare. Per avere una presentazione simile su computer diversi (soprattutto con sistemi operativi diversi) è quindi buona norma specificare sempre una famiglia di font generici da utilizzare nel caso non siano disponibili nessuno di quelli specifici indicati. Il problema della coerenza dei font è abbastanza sentito in quanto sistemi operativi diversi possono mostrare le pagine web in modo molto diverso se non si specificano bene le famiglie di font da utilizzare.

6.8.13 Indicare uno o più font

CSS permette di specificare uno o più font possibili da utilizzare.

```
p { font-family: Verdana, Arial, sans-serif}
```

- I font devono essere separati da virgole
- Nomi dei font-family con caratteri speciali (es.: spazi) devono essere racchiusi tra singoli apici (')
- Si può specificare alla fine una famiglia da cui il browser può attingere i font nel caso non siano presenti quelli elencati

6.9 Altre proprietà impostabili con i CSS

6.9.1 La spaziatura all'interno degli elementi

È possibile controllare gli spazi all'interno degli elementi modificando proprietà che riguardano:

- l'allineamento del testo (text-align)
- l'indentazione del testo (text-indent)
- l'altezza delle linee (line-height)
- gli spazi tra le parole e le lettere (word-spacing, letter-spacing)
- l'allineamento verticale del testo (vertical-align)

6.9.2 Allineamento del testo: text-align

La proprietà `text-align` può assumere i valori:

Esempi di allineamenti	
left (sinistro)	Testo allineato a sinistra
right (destro)	Testo allineato a destra
center (centrato)	Testo centrato
justify (giustificato)	Testo giustificato, mi serve un testo più lungo perché vada a capo :)

Esempio di text-align

```
<p style="text-align: right">
  Pisa, 3 Dicembre 2003
</p>

<p>
  Spett. lettore,
</p>

<p>questa è una possibile
  lettera scritta per fornire
  un esempio di utilizzo della
  proprietà text-align.
</p>

<p style="text-align:right">
  Cordiali saluti</p>
<p style="text-align:right">
  Cristian</p>
```

Pisa, 3 Dicembre 2003

Spett. lettore,
questa è una possibile lettera scritta per fornire un
esempio di utilizzo della proprietà text-align.

Cordiali saluti

Cristian

6.9.3 Indentazione del testo e altezza del righe: text-indent, line-height

La proprietà `text-indent` permette di specificare l'indentazione della prima riga del blocco.

La proprietà `line-height` permette di specificare l'altezza delle righe.

Esempio di text-indent e line-height

```
p {
  text-indent: 2em
}

p {
  line-height: 2em
}
```

Si può indentare la prima riga di ogni paragrafo impostando la proprietà **text-indent**.

Si può aumentare l'interlinea tra le righe di testo adiacenti impostando la proprietà **line-height**.

6.9.4 Spazi tra parole e lettere: word spacing, letter-spacing

Esempi di word spacing e letter-spacing

```
p {
  word-spacing: 2em
}

p {
  letter-spacing: 0.5em
}
```

Si può aumentare lo spazio tra le parole impostando la proprietà **word-spacing**.

Si può aumentare lo spazio tra le singole lettere impostando la proprietà **letter-spacing**.

6.9.5 Allineamento verticale del testo: vertical-align

La proprietà `text-align` permettere di impostare l'allineamento verticale del testo all'interno del box corrente.

vertical-align può assumere i valori:

baseline (la base dell'elemento alla base del padre)	Padre - Testo allineato alla base del padre
sub (nella posizione "a pedice" per il padre)	Padre - Testo allineato al pedice del padre
center (centrato)	Padre - Testo centrato
super (in alto)	Padre - Testo in alto

6.9.6 white-space

La proprietà `white-space` permette di indicare se gli spazi bianchi, gli a capo e le tabulazioni devono essere preservati, cosi come succede nell'elemento HTML `pre`

```
code {
  white-space: pre;
}
```

Su MS-Explorer (<=6) impostare questa proprietà non ha nessun effetto.

6.10 Perché i CSS?

- I CSS permettono di indicare delle regole per la presentazione dei documenti validi per tutti gli elementi dello stesso tipo.
- Le pagine HTML senza gli attributi di presentazione risultano:
  - più leggere (meno codice)
  - più manutenibili (si può modificare la presentazione senza toccare le pagine HTML)
  - indicizzabili in modo migliore (la struttura e il contenuto sono ben separati dalla presentazione)
  - più accessibili (si possono specificare proprietà per client dedicati)

6.10.1 Il (non) problema della retro compatibilità

I CSS sono uno strumento estremamente potente, non possono esserci dubbi. I limiti alla loro diffusione sono essenzialmente due:

- la pigrizia dei webmaster ad apprendere nuove tecniche e a cambiare vecchie (cattive) abitudini;
- la compatibilità con i browser più datati (problema inesistente).

Per definizione, gli standard web, come i CSS e l'XHTML, sono retrocompatibili.

Browser piuttosto datati come IE4 e NN4 hanno una pessima interpretazione dei CSS.

Considerando:

- le ininfluenti statistiche di utilizzo di questi browser
- che l'utilizzo dei CSS favorisce l'accessibilità dei siti web ai diversamente abili

Consigliamo l'utilizzo dei CSS anche se a sfavore di chi utilizza prodotti tecnologicamente obsoleti (che non supportano correttamente i CSS).

Altri elementi HTML

7.1 Collegamenti ipertestuali

In questo paragrafo prenderemo in esame gli ultimi elementi dell'HTML considerati fondamentali per una completa e soddisfacente strutturazione di un documento HTML.

`<a> ... </a>`

Un collegamento ipertestuale è un collegamento tra una risorsa Web ed un'altra; esso ha, quindi, una partenza, una destinazione ed una direzione. La risorsa di arrivo di un collegamento può essere contenuta sia all'interno dello stesso documento, che all'interno di un altro documento che fa parte della stessa collezione, oppure ovunque su Internet.

Un collegamento ipertestuale è identificato dall'elemento **a** che indica una porzione di testo, o un altro oggetto che, se attivato dall'utente, fa si che il Browser si carichi un'altra URI (destinazione). L'oggetto contenuto nell'elemento **a** sarà presentato dal Browser aggiungendo dei particolari effetti per indicare che è un collegamento ipertestuale: tipicamente il testo viene colorato e sottolineato e alle immagini viene aggiunto un bordo.

L'elemento **a** non può contenere alcuni elementi tra cui, elementi di tipo blocco e altre ancore.

E' da tenere presente che l'errore più comune commesso nella realizzazione di un documento HTML è quello di caricarlo di collegamenti ipertestuali utilizzandoli come 'bottoni panico' da selezionare con la classica scritta 'clicca qui'. È consigliabile creare collegamenti più concisi possibile e possibilmente auto esplicativi e consistenti con il documento di arrivo e non mettere mai due collegamenti ipertestuali adiacenti.

Attributi:

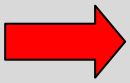
- **href** - Il tag di apertura `<a>` contiene l'attributo `href` necessario per indicare al Browser dove si trova la risorsa Web di destinazione. L'attributo `href` ha come valore la URI di un documento che può essere:
  - assoluta
  - relativa

  Come vedremo nel paragrafo *Attributo "href" e protocolli*, la tipologia del collegamento alla risorsa dipende dalla tipologia dell'indirizzo associato all'attributo `href`.

CSSm-pagina2

Esempio:

```
<a href="http://www.nic.it">
  Sito Web del Registro
</a>
```



Sito Web del Registro

Selezionando "Sito Web del Registro" il Brawser caricherà la pagina contenuta alla URI href://www.nic.it.

- **name** - permette di definire un'etichetta all'interno di un documento la quale può diventare la destinazione di un collegamento ipertestuale.

  Il valore dell'attributo **name** è qualsiasi stringa racchiusa tra virgolette; essa deve essere unica all'interno del documento, ma può essere ripetuta in documenti diversi.

```
<boby>
   ...
   <a href="http://www.favole.it/collana.html#biancaneve">
      Biancaneve
   </a>
   ...
   ...
   <a name="biancaneve"> Biancaneve e i sette nani </a>
   ...
</boby>
```

Selezionando "Biancaneve e i sette nani" il Browser salta all'interno dello stesso documento *collana.html* alla posizione identificata da un'ancora denominata *Biancaneve*.

Le ancore sono spesso utilizzate per facilitare l'accesso alle informazioni contenute in un documento molto lungo. A questo scopo sono utilizzate due metodologie diverse:

Primo metodo:

- il documento è diviso in diverse sezioni che hanno un particolare significato e una formattazione adeguata; ogni sezione è preceduta da un'intestazione consistente con il contenuto;
- ogni intestazione è poi identificata da una etichetta;

- infine, all'inizio del documento, viene costruita una lista di collegamenti ipertestuali alle etichette per facilitare il raggiungimento delle varie sezioni.

Secondo metodo:

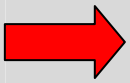
- Il contenuto del documento viene strutturato secondo dei livelli di approfondimento;
- sono creati dei collegamenti ipertestuali da selezionare eventualmente per ottenere informazioni dettagliate su un particolare argomento ed eventualmente tornare indietro per proseguire la lettura del documento originale.

- **`title`** - permette di specificare il titolo della risorsa di destinazione associata al collegamento. È utile quando viene referenziato un oggetto senza nome come un'immagine o un documento che non è HTML[27].

  Tipicamente il valore dell'attributo **`title`** viene visualizzato dal Browser al passaggio del mouse sul collegamento ed è utilizzato come titolo del collegamento nel momento in cui questo viene aggiunto nella lista dei 'preferiti' del Browser.

Esempio:

```
<a
  href="/cgi-bin/StatoReg/"
title="Stato delle registrazioni
del NIC">
  Stato delle Registrazioni
</a>
```



Stato delle Registrazioni

Stato delle Registrazioni del NIC

- **`target`** - l'attributo **`target`** indica il nome della finestra dove sarà visualizzato il documento indicato nell'attributo **`href`**.

  Se non esiste una finestra o un frame con quel nome, il Browser apre una nuova finestra, le da il nome indicato nell'attributo e carica il documento all'interno di essa.

[27] L'attributo `title` può essere usato in tutti gli elementi del body

LaTeX-impaginato

Per questo attributo esistono dei valori predefiniti tra i quali i principali sono:

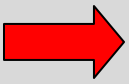
- **_self,** è il valore di default, il documento è visualizzato nella stessa finestra contenente il collegamento;
- **_blank**, il documento è visualizzato ogni volta in un nuova finestra senza nome;
- **_parent**, il documento è visualizzato nella finestra padre se il documento corrente è inserito all'interno di un frame.

Da notare che la stringa indicata come valore dell'attributo target non può iniziare con il carattere '_' che è specifico dei valori predefiniti, qualsiasi altro nome, diverso dai predefiniti, che inizi con il carattere '_' è ignorato dai Browser.

Esempio1:

Apertura di un documento in una nuova finestra:

```
<a href="http://www.nic.it"
   target="_blank"> Sito del NIC
</a>in una nuova finestra
```



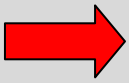
Sito del NIC in una nuova finestra

Selezionando "Sito del NIC" il Browser chiede al server *www.nic.it* la pagina *index.html* che viene visualizzata in una nuova pagina.

Esempio2:

Apertura di un documento nella stessa finestra:

```
<a href="http://www.nic.it"
   target="_self"> Sito del NIC
</a>nella stessa finestra
```



Sito del NIC nella stessa finestra

Selezionando "Sito del NIC" il Browser chiede al server *www.nic.it* la pagina *index.html* che viene visualizzata nella stessa finestra contenente il collegamento ipertestuale.

7.1.1 Attributo "href" e URI relative

Questo paragrafo contiene alcuni esempi di utilizzo dell'attributo href con URI relative.

La URI contenuta nell'attributo href può essere, infatti, assoluta oppure relativa.

Prendiamo il caso in cui l'indirizzo base sia impostato a *http://www.nic.it/RA/news/*:

```
<head>
  ...
  <base href="http://www.nic.it/RA/news/" />
</head>
```

Esempio1:

URI relativa all'indirizzo base (stesso livello):

La URI: `<a href="index.html"> indice1 </a>`

viene tradotta come: `http://www.nic.it/RA/news/index.html`

Esempio2:

URI relativa all'indirizzo base (livello precedente):

La URI: `<a href="../index.html"> indice2 </a>`

viene tradotta come: `http://www.nic.it/RA/index.html`

Esempio3:

URI relativa all'indirizzo base (livello precedente):

La URI: `<a href="/index.html"> indice2 </a>`

viene tradotta come: `http://www.nic.it/index.html`

LaTeX-impaginato

7.1.2 Attributo "href" e protocolli

Questo paragrafo mostra alcuni esempi di utilizzo dell'attributo href contenenti URI con protocolli diversi.

Esempio1:

URI con protocollo http:

```
<a href="http://www.nic.it/RA/index.html"> Sito Web del NIC </a>
```

Sito Web del NIC

Selezionando con il mouse "**Sito del NIC**" il Browser richiede al server *www.nic.it* il documento *index.html* che sarà visualizzato nella finestra del Browser.

Esempio2:

URI con protocollo ftp:

```
<a href="ftp://ftp.nic.it/RA/doc/Modulo-Registrazione-Dominio.txt">
      Modulo di Registrazione di un nome a dominio
</a> …
```

Modulo di Registrazione di un nome a dominio …

Selezionando "**Modulo di Registrazione di un nome a dominio**" il Browser contatta il server *ftp.nic.it* per l'invio del file, chiede conferma per lo scaricamento del file *Modulo-Registrazione-Dominio.txt* ed eventualmente il file viene scaricato sulla macchina[28].

[28] Se è un formato conosciuto, il Browser può aprire il file direttamente senza chiedere conferma

Esempio3:

URI con protocollo mailto:

```
Per informazioni contattare il <a href="malto:webmaster@nic.it">
webmaster </a> del NIC
```

Per informazioni contattare il webmaster del NIC

Selezionando con il mouse "**webmaster**" il Browser attiva un programma di posta elettronica e apre una e-mail vuota con destinatario webmaster@nic.it.

7.2 Tabelle

`<table> … </table>`

Una tabella è racchiusa tra i tag di apertura e di chiusura `<table>` e `</table>` all'interno dei quali sono inseriti tutti gli elementi che definiscono le righe e le colonne della tabella stessa.

Di default, quando incontra una tabella, un Browser grafico ha il seguente comportamento:

a. interrompe il flusso di testo corrente;
b. inserisce la tabella all'inizio di una riga nuova;
c. ricomincia il flusso di testo dopo la tabella su una riga nuova.

Una tabella inserita nel corpo di un documento HTML può contenere qualsiasi cosa: immagini, testo, form, intestazioni e persino altre tabelle. Il Browser ridimensiona automaticamente la larghezza e la lunghezza della tabella sulla base del suo contenuto.

Il Browser tratta le celle di una tabella come se fossero singole finestre all'interno della finestra principale: il contenuto della cella è distribuito fino a riempire lo spazio intero e per esso è possibile definire uno stile di presentazione appropriato.

All'interno di un documento, le tabelle devono essere utilizzate per rappresentare dei dati e non, come spesso accade, come elementi di formattazione del documento stesso. Un documento formattato tramite l'utilizzo di una tabella può risultare poco accessibile e usabile a seconda del client in cui viene rappresentato il documento (Es: client non grafici o client grafici con display piccoli).

Un documento contenente molte tabelle, specialmente se annidate, richiede una massiccia elaborazione del browser per la presentazione. Il Browser deve, infatti, calcolare il numero delle celle e successivamente ridimensionarle una per una a seconda del loro contenuto; se il contenuto di una cella è a sua volta una tabella, l'elaborazione deve essere ripetuta.

7.2.1 Definizione delle celle

<tr> ... </tr>, **<td> ... </td>**, **<th> ... </th>**

All'interno di una tabella, ogni riga è definita dai tag **<tr> … </tr>**.

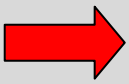
Ogni riga può contenere una o più celle definite da elementi diversi a seconda del loro contenuto:

- **<th> .. </th>**, se la cella contiene il titolo o la descrizione di una colonna;
- **<td> .. </td>**, se la cella contiene dati.

Il Browser rappresenta il contenuto delle celle in modo differenziato a seconda della loro definizione; in un Browser testuale il contenuto della cella **<th>** viene solitamente visualizzato in grassetto e con allineamento centrato.

Esempio:

```
<table>
<tr>
  <th> Nome </th>
  <th> Cognome </th>
</tr>
<tr>
  <td> Mario </td>
  <td>Rossi </td>
</tr>
</table>
```



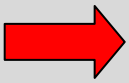
Nome	Cognome
Mario	Rossi

Ogni riga della tabella ha lo stesso numero di celle, definito dalla riga più 'lunga'.

Se una riga contiene un numero di celle minore, il Browser crea automaticamente delle celle vuote per riempirne lo spazio.

Esempio:

```
<table>
<tr>
  <th> Nome </th>
  <th> Cognome </th>
</tr>
<tr>
  <td> Mario </td>
</tr>
</table>
```



Nome	Cognome
Mario	

7.2.2 Definizione del contenuto

`<caption … />`

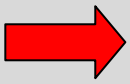
È buona norma fornire sempre una tabella di una sua descrizione.

L'elemento **`caption`** consente di aggiungere alla tabella una breve descrizione del suo contenuto; il testo viene visualizzato dai Browser grafici prima o dopo la rappresentazione del contenuto della tabella stessa. Una tabella dovrebbe sempre contenere l'elemento **`caption`** che deve essere inserito appena dopo il tag di apertura **`<table>`**. Inoltre, è permesso un solo elemento **`caption`** all'interno di una tabella.

Un altro modo di aggiungere alla tabella una descrizione è quello di utilizzare l'attributo **`summary`** all'interno del tag di apertura **`<table>`**; esso consente di inserire una descrizione più approfondita sia sul contenuto della tabella che sulla sua struttura. L'attributo **`summary`** è utile per client non grafici come quelli vocali o Braille.

Esempio:

```
<table summary="questa tabella
fornisce la distribuzione dei
registranti dei nomi a dominio per
sesso e classi di età.
 Dalla tabella si evince che vi sono
 più donne nelle classi di età più
 elevate e che la classe tra 18- 41 è
 più rappresentata sia per gli uomini
 sia per le donne">

<caption> <em>Distribuzione
registranti nomi a dominio per sesso e
per classi di età</em>
</caption>

<tr>
  <th> </th>
  <th> 18-41  </th>
  <th> 42-65  </th>
  <th> Oltre 65 </th>
</tr>

<tr>
  <th> Donne </th>
  <td>66,3 </td>
  <td>30,5 </td>
  <td>3,1  </td>
</tr>

<tr>
  <th> Uomini </th>
  <td>71,7 </td>
  <td>26,8 </td>
  <td>1,5 </td>
</tr>

</table>
```



Distribuzione registranti nomi a dominio per sesso e per classi di età

	18-41	42-65	Oltre 65
Donne	66,3	30,5	3,1
Uomini	71,7	26,8	1,5

Attributi:

In HTML è possibile definire celle che prendono lo spazio di due o più colonne e/o righe all'interno della tabella; questo è possibile attraverso l'utilizzo degli attributi `colspan` e `rowspan`:

- `colspan="n"`, indica al browser di far occupare, alla cella corrente, lo spazio di 'n' colonne della stessa riga

- `rowspan="n"`, indica al browser di far occupare, alla cella corrente, lo spazio di 'n' righe della stessa colonna

Gli attributi `colspan` e `rowspan` possono essere combinati per disporre una singola cella su più colonne e/o righe contemporaneamente; il valore di default è 1.

Cosa accade se nella tabella non esistono abbastanza celle per espandere la cella attuale? Il Browser espande la cella indicata sulle celle disponibili, nella riga o nella colonna, senza aggiungerne di nuove.

Esempio:

```
<table summary="questa tabella fornisce
 la distribuzione dei registranti dei
 nomi a dominio per sesso e classi di
 età.
 Dalla tabella si evince che vi sono
 più donne nelle classi di età più
 elevate e che la classe tra 18- 41 è
 più rappresentata sia per gli uomini
 sia per le donne">

<caption> <em>Distribuzione registranti
 nomi a dominio per sesso e per classi
 di età  </em>
</caption>

<tr>
  <th rowspan="2"></th>
  <th colspan="3">
    Distribuzione per sesso
  </th>
</tr>

<tr>
  <th> 18-41  </th>
  <th> 42-65  </th>
  <th> Oltre 65 </th>
</tr>

<tr>
  <th> Donne </th>
  <td>66,3 </td>
  <td>30,5 </td>
  <td>3,1  </td>
</tr>

<tr>
  <th> Uomini </th>
  <td>71,7 </td>
  <td>26,8 </td>
  <td>1,5 </td>
</tr>

</table>
```

Distribuzione registranti nomi a dominio per sesso e per classi di età

	Distribuzione per sesso		
	18-41	42-65	Oltre 65
Donne	66,3	30,5	3,1
Uomini	71,7	26,8	1,5

7.2.3 Gli elementi colgroup e col

```
<colgroup … />
<col />
```

L'elemento **colgroup**[29] identifica un gruppo di colonne all'interno di una tabella con due obiettivi:

1. Identificare e definire un insieme di colonne con stesse proprietà
   - l'attributo **span="n"** definisce il numero di colonne interessate dalle proprietà definite nell'attributo
2. Raggruppare un insieme di colonne con diverse proprietà
   - All'interno del tag **<colgroup>**, per ogni colonna appartenente al gruppo, viene inserito un tag **<col>** all'interno del quale sono definite le proprietà della colonna; in questo caso l'attributo **span** non viene utilizzato.

Esempio1:

```
<colgroup span="2" />
<colgroup span="1" />
```

Esempio2:

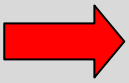
```
<colgroup>
  <col />
  <col />
</colgroup>
<colgroup>
  <col />
</colgroup>
```

[29] L'elemento **colgroup** è supportato da Internet Explorer e Netscape 6;

Gli elementi di definizione delle colonne devono essere inseriti subito dopo il tag di apertura **<table>** e prima della definizione del contenuto della tabella. Inoltre, a differenza degli altri elementi utilizzati per la definizione delle tabelle, i tag **<colgroup>** e **<col>** non possono racchiudere il contenuto della tabella stessa.

Esempio1:

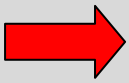
```
<table>
  <colgroup span="2" width="20%" />
  <colgroup span="2" width="30%" />
  <tr>
    <th>Donne</th>
    <td>66,3</td>
    <td>30,5</td>
    <td>3,1</td>
  </tr>
  <tr>
    <th>Uomini</th>
    <td>71,7</td>
    <td>26,8</td>
    <td>1,5</td>
  </tr>
</table>
```



Donne	66,3	30,5	3,1
Uomini	71,7	26,8	1,5

Esempio2:

```
<table>
  <colgroup>
         <col width="20%" />
         <col width="20%" />
  </colgroup>
  <colgroup>
         <col width="30%" />
         <col width="30%" />
  </colgroup>
  <tr>
    <th>Donne</th>
    <td>66,3</td>
    <td>30,5</td>
    <td>3,1</td>
  </tr>
  <tr>
    <th>Uomini</th>
    <td>71,7</td>
    <td>26,8</td>
    <td>1,5</td>
  </tr>
</table>
```



Donne	66,3	30,5	3,1
Uomini	71,7	26,8	1,5

Le immagini

8

8.1 Inserire immagini

Una delle caratteristiche più attraenti dell'HTML è la capacità di inserire immagini all'interno di un documento di testo, sia come parti integranti e in linea del documento, sia come documenti separati e selezionabili per lo scaricamento via hyperlink, sia infine come sfondo del documento.

Le immagini, se utilizzate con criterio all'interno del documento sotto forma di icone (sia statiche che dinamiche), di foto, di illustrazioni, di disegni ed altro ancora, possono rendere il documento da un lato più attraente e dall'aspetto più professionale, dall'altro più istruttivo e facile da sfogliare. In particolare è possibile trasformare un'immagine in una mappa “sensibile”, vale a dire un'area visibile sulla quale definire una serie di collegamenti ipertestuali.

Al contrario, se usate scriteriatamente, le immagini possono rendere il documento ingombrante, confuso e inaccessibile, prolungando senza necessità il tempo necessario allo scaricamento e alla visualizzazione delle pagine.

CSSm-pagina2

8.2 Formati di immagine

Né l'HTML né l'XHTML prescrivono nelle rispettive specifiche un formato “ufficiale” da utilizzare per le immagini, anche se i browser più popolari (che spesso e volentieri impongono degli standard di fatto) hanno sempre favorito fino dalle versioni più remote l'utilizzo di GIF (Graphics Interchange Format) e JPEG (Joint Photographic Experts Group), gestendo tali formati a livello nativo.

Al GIF e al JPEG si è aggiunto abbastanza di recente (specifica W3C dell'ottobre 1996) il PNG (Portable Network Graphics), nato per trovare un sostituto non proprietario al formato GIF (alla fine del 1994 la società UNISYS aveva deciso di richiedere le royalties a tutti gli sviluppatori di software grafico che supportasse tale formato).

Comunque sia, tutti i principali applicativi grafici supportano i formati citati, permettendo la loro creazione e modifica. E ciascuno di essi presenta vantaggi e svantaggi che li rende più o meno adatti alla tipologia di immagine da rappresentare.

8.2.1 Graphics Interchange Format (GIF)

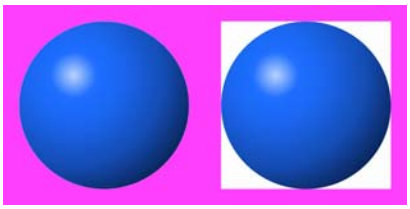
Il Graphics Interchange Format (GIF) fu sviluppato inizialmente per lo scambio ed il trasferimento di immagini tra utenti del servizio online CompuServe. Questo formato ha diverse caratteristiche che lo hanno reso popolare per un utilizzo in HTML e XHTML. La sua codifica è indipendente dalla piattaforma software e quindi un opportuno decodificatore software, incluso nella maggior parte dei browser, può scaricare e visualizzare su un computer Windows un grafico GIF creato su Macintosh.

La seconda caratteristica importante del formato GIF è la speciale tecnologia di compressione, che permette considerevoli risparmi di dimensioni per un agevole trasferimento sulla rete. Inoltre l'algoritmo di compressione è senza perdita di dati (“lossless”), vale a dire che nella compressione nessuna parte dell'immagine viene alterata o persa per strada, facendo si che una volta decompressa e decodificata dal browser, essa sia identica all'originale.

Anche se l'immagine GIF ha sempre come estensione di nome file ".gif" o ".GIF", attualmente esistono due versioni del formato, il GIF87 originale e il GIF89 che gestisce caratteristiche più avanzate quali l'interlacciamento (o visualizzazione progressiva), gli sfondi trasparenti e le animazioni.

La tecnica dell'interlacciamento permette una visualizzazione dell'immagine più immediata anche se all'inizio meno risoluta, dal momento che dalla matrice di pixel che compongono l'immagine, viene prelevata e visualizzata solo una riga su quattro. L'utente, in un quarto del tempo, ha una visualizzazione abbastanza chiara (anche se a “tendina veneziana”) dell'immagine che sta scaricando, e può prendere le opportune decisioni: interrompere lo scaricamento o continuare, nel qual caso con altre tre passate successive si ottiene la visualizzazione definitiva e a piena risoluzione dell'immagine. Si tratta comunque di un guadagno più “psicologico” che reale, dato che il tempo totale di scaricamento delle quattro passate coincide (se non è leggermente superiore) a quello dello scaricamento della stesa immagine non interlacciata.

Un altra caratteristica molto usata del formato GIF è quella della trasparenza. L'immagine GIF trasparente ha un colore tra quelli della sua “palette” che viene definito trasparente, e pertanto tutti i pixel dell'immagine di quel colore non vengono visualizzati dal browser, permettendo di intravedere ciò che è presente al di sotto del rettangolo che dimensiona l'immagine. Definendo opportunamente il pixel trasparente (e se il contenuto dell'immagine si presta a questa trasformazione) è possibile inserire in maniera più realistica le immagini all'interno del flusso della pagina del documento.



La sfera a sinistra ha il colore bianco definito trasparente, la destra no

La terza caratteristica singolare del formato GIF89a è quella di gestire semplici animazioni, create con appositi software. È possibile preparare un singolo file GIF89a che contiene una serie di GIF successive che, una volta riprodotte sul browser, danno l'idea del movimento.

Serie di immagini diverse per simulare l’animazione

Di contro, a causa del limitato numero massimo di colori disponibili (256, 8 bit), il formato GIF non si presta per la rappresentazione di immagini fotorealistiche. È idoneo piuttosto per la realizzazione di logo, icone e grafici, tutte immagini con un ridotto numero di colori.

8.2.2 Joint Photographic Experts Group (JPEG)

Il formato Joint Photographic Experts Group (JPEG), come il GIF, è indipendente dalla piattaforma software ed ha un eccellente algoritmo di compressione (da 10:1 a 100:1), particolarmente indicato per trasferire in rete immagini all'origine complesse.

Diversamente dal formato GIF, il JPEG gestisce fino a 16.777.216 colori (2 alla 24), che rendono altamente realistiche e dettagliate le immagini digitali.

Lo speciale algoritmo di compressione può portare immagini GIF di 200 KB a immagini JPEG di 30 KB. Purtroppo ciò presuppone la perdita, durante la compressione, di parte dei dati originali, che non saranno più recuperati durante la decompressione sul browser. È comunque possibile decidere la grandezza di tale perdita, regolando il fattore di compressione: più basso tale fattore, minore perdita di dati, maggiore qualità, maggiori dimensioni. Aumentando il fattore di compressione, si ottiene maggiore perdita di dati, minore qualità, minori dimensioni. Con un opportuno bilanciamento, si riesce comunque ad ottenere immagini JPEG di ridotte dimensioni e con perdita di qualità non avvertibile all'occhio umano.

Bisogna fare grande attenzione al fatto che la perdita dei dati occorre ad ogni successivo salvataggio del file, per cui conviene sempre utilizzare un formato "lossless" per la modifica dell'immagine, e fare un solo salvataggio finale per la versione da pubblicare su documento WEB.

8.2.3 Portable Network Graphics (PNG)

Il formato PNG (Portable Network Graphics), presenta praticamente tutti i vantaggi dei formati GIF e JPEG, con addirittura alcune peculiarità proprie: dispone infatti di una trasparenza su 256 livelli, può memorizzare informazioni testuali con l'immagine stessa, e infine gestisce l'adattamento automatico di luminosità e contrasto a seconda della piattaforma usata.

Di norma, la stessa immagine visualizzata su un Macintosh apparirà molto più chiara su un personal computer Windows. Memorizzando nell'immagine PNG l'informazione sulla correzione gamma, la visualizzazione verrà resa nella stessa maniera sulle piattaforme diverse.

In definitiva, il formato PNG rappresenta, rispetto al GIF, un sostituto non coperto da brevetto a pagamento, portabile, senza perdita di dati, con trasparenza, con profondità di colore fino a 48 bit (281.474.976.710.656

colori per pixel) e con una compressione dal 5% al 25% migliore del formato GIF.

Gli unici due fattori contrari sono il mancato supporto da parte dei browser più vecchi e la mancanza della possibilità di animazione.

8.3 Quando usare immagini

Le immagini dovrebbero essere di supporto al contenuto testuale, aiutando colui che legge a navigare all'interno del documento.

Fotografie, tavole, grafici, mappe, e disegni in genere che supportano la parte testuale: sono tutti naturali ed appropriati candidati all'uso come immagine. Foto di prodotti sono componenti insostituibili nel caso di cataloghi online e guide per l'acquisto. E icone o piccole immagini animate possono essere una buona guida visuale per indirizzare verso risorse interne ed esterne. Se un'immagine non soddisfa questi requisiti, tiratela pure via dalla vostra pagina.

Una delle considerazioni più importanti da fare nell'aggiungere immagini al documento è che esse incrementano considerevolmente il tempo di scaricamento sulla rete, specie in caso di connessioni lente via modem. Mentre un comune documento di testo al massimo è composto da 10 - 15 mila bytes, le immagini possono facilmente estendere la dimensione a centinaia di migliaia di bytes per ciascuna. E considerate che il tempo totale di scaricamento di un documento non è formato solo dalla somma aritmetica dei tempi necessari a tutti i suoi componenti (testo più immagini), ma è progressivamente incrementato anche dalla complessità stessa del documento.

A seconda della velocità della connessione (o larghezza di banda) espressa in bit o byte per secondo, e a seconda della congestione della rete dovuta all'orario del collegamento, un documento con una sola immagine di 100 KB può impiegare circa 15 secondi con un modem a 57.6 KBps nelle prime ore del mattino (a rete scarica), fino a oltre 10 minuti con un modem a 9600 bps verso sera.

8.4 Quando usare testo

Nonostante le pagine WEB siano sempre più ricche di immagini e/o animazioni, il ricorso a documenti formati da solo testo non sono certo anacronistici. Intanto, per determinati utenti, il testo è l'unica porzione del documento che possono accedere. Il documento dovrebbe perciò essere "usabile" anche dagli utenti che non possono vedere le immagini o che hanno volutamente disabilitato il loro scaricamento automatico.

Anche se la necessità di aggiungere immagini al testo può essere notevole, ci sono circostanze nelle quali documenti solamente testuali hanno più senso: materiale convertito da altri formati per il Web e documenti di consultazione raramente hanno immagini inserite all'interno, rimanendo pur tuttavia perfettamente utilizzabili anche in formato solamente testuale.

Considerate inoltre che il materiale testuale è più appropriato se i documenti devono essere prontamente ricercati dai servizi di indicizzazione sul Web, mentre le immagini sono solitamente ignorate dai motori di ricerca. Se la maggior parte del contenuto delle vostre pagine è rappresentato da immagini, ben poche informazioni sui vostri documenti saranno inserite negli appositi elenchi informativi in linea sulla rete.

8.5 Velocizzare lo scaricamento delle immagini

Un'immagine a pieno schermo con 24 bit di profondità di colore (16 milioni e passa), anche se ridotta con compressione digitale da uno dei formati standard (GIF, JPEG o PNG) richiede sempre una notevole occupazione di banda. È perciò necessario ottimizzare dimensioni e numero di colori già in fase di acquisizione e/o creazione e modifica. Ciò significa semplificare i disegni, evitare fotografie panoramiche così come grandi sfondi vuoti o bordi non necessari. Evitate ove possibile il "dithering" (mescolare insieme due colori di pixel adiacenti per ottenerne un terzo), in quanto questa tecnica riduce significativamente la possibilità di comprimere l'immagine. Cercate piuttosto di realizzare grosse parti di colore uniforme, che vengono compresse in maniera ottimale dagli algoritmi di tutti i formati di immagine.

Prevedete un link speciale per grosse immagini, magari con una rappresentazione in miniatura e l'informazione delle dimensioni, lasciando all'utente la decisione se e quando scaricare l'immagine piena.

Infine, fornire sempre nel tag di inserimento dell'immagine, le informazioni sulla larghezza e sull'altezza. Quando il browser incontra un elemento <img>, deve predisporre sulla pagina che sta formattando l'apposito riquadro. Se forniamo le due misure dell'altezza e dell'ampiezza, evitiamo che il browser debba andare a ricavare tali dati esaminando il contenuto dell'immagine.

8.6 L'elemento `img`

`<img>`

Funzione

Inserisce un'immagine all'interno di un documento

Attributi

`alt, height, ismap, longdesc, src, usemap, width`

Contiene

Nulla

Utilizzo

Nel flusso del testo

Il formato con cui l'immagine è codificata non è definito da nessuno standard HTML o XHTML, benché i browser più popolari supportano immagini in formato GIF, JPEG e PNG, e benché il W3C Consortium spinga fortemente per l'utilizzo del formato PNG. Gli standard non specificano o impongono restrizioni sulla grandezza, le dimensioni o il numero dei colori delle immagini (come poi tali colori vengano visualizzati dipende completamente dal browser e dalle possibilità grafiche del computer su cui visualizziamo le immagini.

8.6.1 L'attributo `src`

- l'attributo **`src`** è obbligatorio;
- il suo valore indica l'URI del file dell'immagine.

In questi esempi di codice e in quelli che seguono, per chiarezza e semplicità, non sempre sono riportati tutti gli attributi, anche se obbligatori.

L'attributo **`src`** dell'elemento **`img`** è obbligatorio, a meno di usare l'attributo **`dynsrc`** con immagini visualizzate su Internet Explorer.

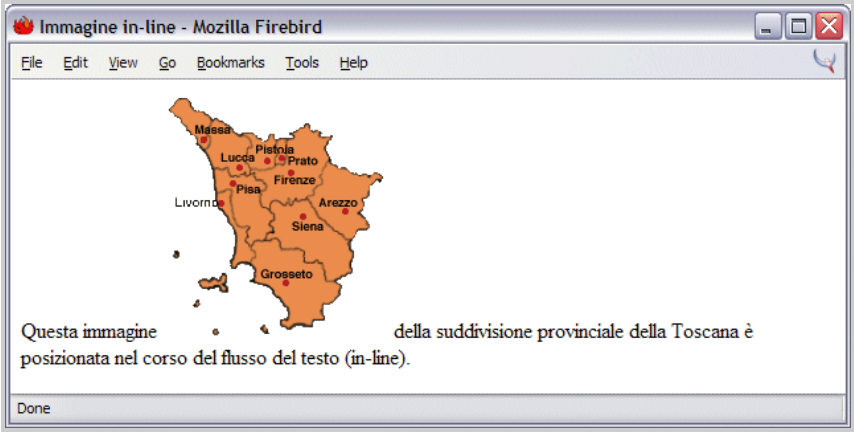
Il suo valore rappresenta l'URI del file dell'immagine, espresso o in maniera assoluta (http://percorso/images/nome_immagine.png) o in maniera relativa al documento che referenzia l'immagine (/images/nome_immagine.png).

Per tenere ordinati logicamente i file su disco, generalmente gli autori raccolgono i file di immagini in cartelle separate chiamate solitamente "pics" o "images".

L'attributo src (esempio 1)

Questo frammento di codice HTML posiziona un'immagine all'interno della frase di testo:

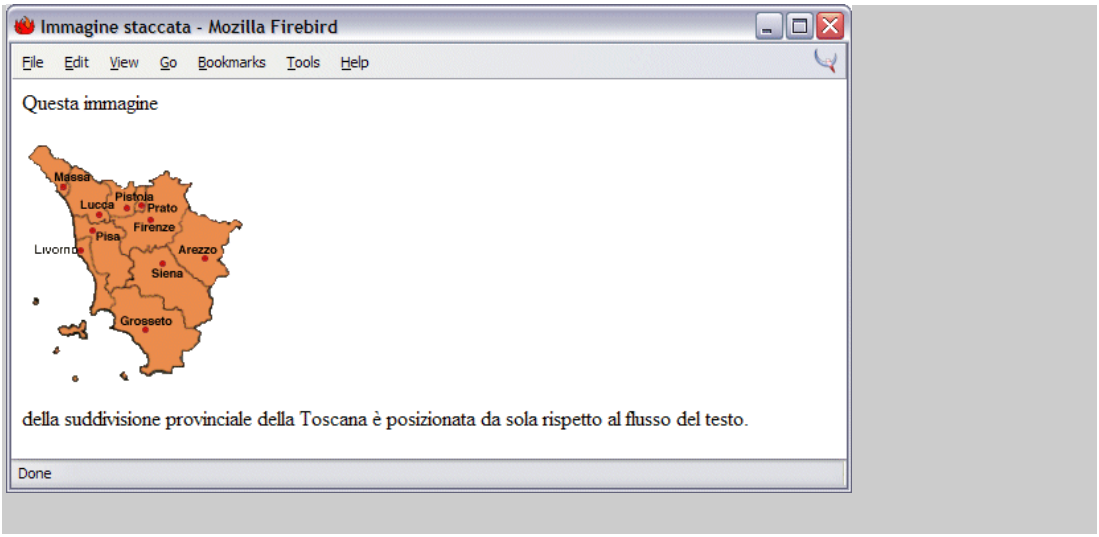
```
<p>Questa immagine <img src="images/toscana.gif" />
della suddivisione provinciale della Toscana &egrave; posizionata
nel corso del flusso del testo (in-line).</p>
```



L'attributo src (esempio 2)

Inserendo l'elemento **`img`** tra un'apertura e una chiusura di un **`div`** l'immagine viene visualizzata dal browser da sola, con dello spazio sopra e sotto:

```
<div>Questa immagine
  <div><img src="images/toscana.gif" /></div> della suddivisione
  provinciale della Toscana &egrave; posizionata da sola rispetto
  al flusso del testo.
</div>
```



8.6.2 L'attributo `alt`

L'attributo **`alt`** specifica un testo alternativo che il browser può mostrare tramite una finestrella di aiuto se la visualizzazione dell'immagine è impossibile o è disabilitata dall'utente. Mediante questo attributo, se l'immagine non è disponibile, l'utente ha comunque qualche indicazione su cosa manca. E per alcuni utenti disabili l'attributo **`alt`** è l'unico modo per referenziare l'immagine.

La finestrella gialla di aiuto è solo un'aggiunta grafica e non tutti i browser grafici la fanno vedere, mentre tutti i browser non grafici utilizzano il valore dell'attributo per specificare in qualche il modo il contenuto dell'immagine.

Il valore dell'attributo è una stringa testuale grande fino a 1.024 caratteri, compresi spazi e punteggiatura. La stringa deve essere inclusa tra apici, e non può contenere altri comandi di marcatura (in particolare tag di stile).

L'attributo alt (esempio)

Questo frammento di codice HTML fa visualizzare il "testo alternativo" quando si passa col mouse sull'immagine:

```
<p>Questa immagine <img src="images/toscana.gif"
alt="Suddivisione provinciale della regione Toscana" />
della suddivisione provinciale della Toscana presenta un testo
alternativo.</p>
```

8.6.3 Gli attributi `height` e `width`

Mai notato come la visualizzazione del contenuto di una pagina si sposti continuamente e in maniera irregolare mentre il documento viene scaricato dalla rete? Il motivo è che il browser risistema il layout per adattarlo a ciascuna immagine che man mano viene caricata. Le dimensioni dell'immagine (e conseguentemente il rettangolo da predisporre per essa sulla pagina, sono estratte dalle specifiche di altezza e di ampiezza contenuto all'interno del file stesso.

Questo non è certo il modo migliore per presentare un documento su un browser, perchè richiede più tempo e confonde il lettore. Se invece specifichiamo i due attributi **`height`** e **`width`**, il browser riserva lo spazio necessario prima di iniziare a caricare l'immagine, velocizzando così la presentazione della pagina ed evitando il fenomeno dello spostamento continuo.

Uno svantaggio di specificare a priori le dimensioni è ... che queste vanno conosciute: il browser riserva sempre lo spazio come dichiarato con gli attributi **`height`** e **`width`**, e in caso di difformità tra le dimensioni reali e quelle dichiarate, l'immagine viene presentata deformata (stirata o contratta). Ciò però offre anche la possibilità di creare delle versioni francobollo delle immagini reali, e la possibilità di riempire larghe aree della pagina con lo stesso colore utilizzando un'immagine di un solo pixel, stirata

in lungo e in largo quanto si vuole, con ovvi risparmi in termini si scaricamento dalla rete.

Gli attributi height e width (esempio 1)

Questo frammento di codice HTML dimostra come la specifica degli attributi `height` e `width` "forzi" le dimensioni dell'immagine:

```
<img src="images/toscana.gif" height="150" width="220" />
```

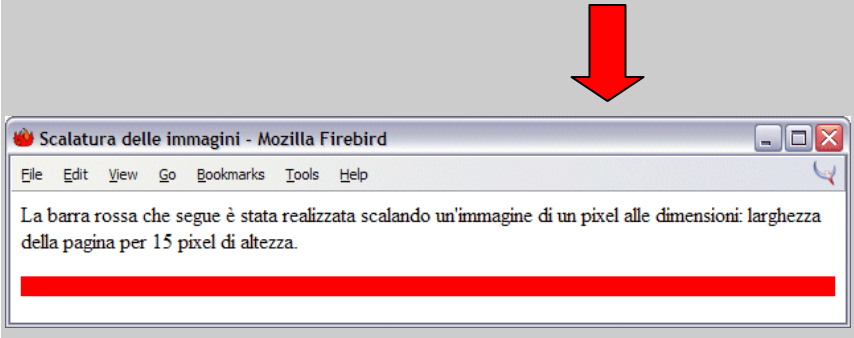
L'immagine della suddivisione provinciale della Toscana risulta distorta perchè sono stati specificati i due attributi `height` e `width` con valori diversi dalle effettive dimensioni originali (171 x 187).

Gli attributi height e width (esempio 2)

Questo frammento di codice HTML dimostra come inserire una barra colorata larga tutta la pagina utilizzando un'immagine grande 1 solo pixel:

La barra rossa che segue è stata realizzata scalando un'immagine di un pixel alle dimensioni: larghezza della pagina per 15 pixel di altezza.

```
<img src="images/one_red_pixel.gif" width="100%" height="15" />
```



8.6.4 Gli attributi `ismap` e `usemap`

Gli attributi **`ismap`** e usemap dell'elemento **`img`** informano il browser che l'immagine è di un tipo speciale, selezionabile col mouse su una o più zone corrispondenti ad altrettanti collegamenti ipertestuali.

La versione ismap delle mappe sensibili, conosciuta anche come server-side, può essere specificata solo all'interno di un elemento **`a`**.

Quando l'utente fa clic sulla immagine ismap, il browser invia automaticamente al server le coordinate x e y (relative all'angolo in alto a sinistra dell'immagine). Uno speciale software sul lato server utilizzerà queste informazioni numeriche per prendere le opportune iniziative.

La versione **`usemap`**, conosciuta anche come client-side, fornisce un meccanismo che elimina effettivamente l'elaborazione delle coordinate del mouse da parte del server, con i ritardi e le problematiche connesse. Oltretutto, non richiedendo software particolari lato server, le mappe sensibili client-side possono essere usate anche in un ambito non Web (o comunque off-line), come file locali o applicazioni su CD-ROM.

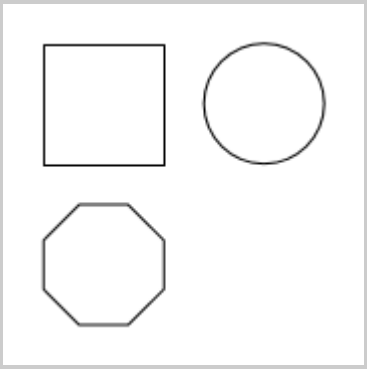
Lo stile **`usemap`** utilizza gli speciali elementi **`<map>`** e **`<area>`** ed è compito dell'autore della pagina fornire una mappa di coordinate relativa all'immagine e gli URI o le azioni da compiere.

Il valore dell'attributo **`usemap`** è un URI che punta alla speciale sezione **`<map>`**.

Gli attributi ismap e usemap (esempio)

Questo frammento di codice HTML fa visualizzare un'immagine con tre forme che, sensibili al passaggio del mouse, impostano un messaggio diverso sulla barra di stato del browser:

```
<img src="images/esempio_usemap_1.gif" usemap="#forme" />
<map name="forme">
  <area shape="rect" coords="20,20,80,80"
    onmouseover="self.status='Sei col mouse sul quadrato.'"
    onmouseout="self.status='Non sei su nessuna forma.'">
  <area shape="circle" coords="130,50,30" nohref="nohref"
    onmouseover="self.status='Sei col mouse sul cerchio.'"
    onmouseout="self.status='Non sei su nessuna forma.'">
  <area shape="poly" coords="38,100,62,100,80,118,80,142,
                             62,160,38,160,20,142,20,118"
    onmouseover="self.status='Sei col mouse sull´ottagono.'"
   onmouseout="self.status='Non sei su nessuna forma.'">
</map>
```



8.7 Le immagini sensibili al mouse

- di norma un'immagine posta all'interno di un elemento a diviene parte del contenuto del link;
- l'utente può cliccare l'immagine allo stesso modo di un link testuale;
- gli standard HTML e XHTML forniscono un sistema per inserire diversi link all'interno della stessa immagine;
- cliccando diverse aree dell'immagine si ottiene il link a diversi documenti (o a diverse azioni);
- esistono immagini sensibili "server-side" e "client-side";

Abbiamo introdotto il concetto di immagini sensibili al mouse parlando dei due attributi `ismap` e `usemap` dell'elemento `img`.

Vediamo ora più in dettaglio come funzionano le mappe sensibili. Di norma, un'immagine posta all'interno di un collegamento ipertestuale, diviene semplicemente parte del contenuto del collegamento. Il browser può modificare l'immagine in qualche modo (con un bordo colorato) per avvertire l'utente che si tratta di un collegamento, e come ogni link ipertestuale il navigatore raggiunge quel collegamento semplicemente facendo clic col mouse.

Con le mappe sensibili gli standard HTML e XHTML forniscono la possibilità di inserire più link differenti all'interno della stessa immagine, cosicché cliccando differenti zone dell'immagine, si salta a differenti documenti di destinazione.

8.7.1 Immagini sensibili "server-side"

Per rendere sensibile un'immagine posta all'interno di un elemento a basta aggiungere l'attributo `ismap`. Questo attributo informa il browser che l'immagine è una mappa speciale contenente più di un link

Quando l'utente clicca in qualche posizione sull'immagine, il browser passa al server le coordinate del puntatore del mouse insieme all'URI specificato nell'elemento a. Il server utilizza tali coordinate per determinare quale documento spedire indietro al browser. Le coordinate della posizione del mouse sono valori di pixel dello schermo relativi all'angolo in alto a sinistra (0,0). Le coordinate, precedute da un punto di domanda, vengono aggiunte alla fine dell'URI.

Per esempio, se l'utente clicca 43 pixels a destra e 15 pixels in basso rispetto all'angolo alto a sinistra dell'immagine visualizzata in base all'URI `<a href="/cgi-bin/imagemap/toolbar.map"> <img ismap src = "pics/toolbar.gif"> </a>`, il browser invia i seguenti parametri di ricerca al server HTTP: /cgi-bin/imagemap/toolbar.map?43,15...

8.7.2 Immagini sensibili "client-side"

Ovviamente è implicito che le immagini sensibili server-side richiedano un server. È necessario allora accedere il server o la sua directory /cgi-bin, entrambi raramente accessibili a tutti. Oltretutto viene diminuita la portabilità della pagina, poichè non tutte le applicazioni che processano le immagini server-side sono uguali. Senza pensare al ritardo per l'utente mente naviga, dal momento che il browser deve ricevere la disponibilità del server a processare le coordinate (anche se viene cliccata una zona non sensibile dell'immagine!).

Le immagini client-side non sono sottoposte questo tipo di problematiche. Abilitate dall'attributo `usemap` dell'elemento `img` e definite dagli speciali tag di estensione `<map>` e `<area>`, le immagini client-side permettono all'autore di includere nei loro documenti mappe di coordinate e link ipertestuali che descrivono le regioni sensibili di un'immagine.

Il browser sul computer cliente traduce le coordinate della posizione del mouse sull'immagine in una determinata azione (ad esempio il caricamento di un altro documento) e speciali attributi processati dal linguaggio JavaScript possono produrre tutta una serie di effetti speciali.

Per creare una mappa sensibile client-side, bisogna inserire l'attributo `usemap` come parte dell'elemento `img`. Il suo valore è l'URI di un segmento `<map>` in un documento HTML che contiene le coordinate della mappa ed i relativi link. Il documento nell'URI identifica il documento HTML che contiene la mappa e la porzione di identificazione identifica la mappa da utilizzare.

Molto spesso la mappa delle coordinate risiede nello stesso documento dove è contenuta l'immagine, e l'URI si riduce così alla sola porzione di identificazione: il cancelletto (#) seguito dal nome della mappa. Ad esempio il seguente frammento di codice dice al browser che l'immagine *map.gif* è una mappa sensibile client-side e che le coordinate delle zone sensibili e i relativi link sono referenziati nella sezione `mappa` del documento stesso.

```
<img src="pics/map.gif" usemap="#mappa">
```

8.8 L'elemento `map`

`map`

Funzione

Racchiude le specifiche di un'immagine sensibile "client-side" (usemap)

Attributi

`class, id, name, style, title`

Contiene

Il contenuto della mappa sensibile

Utilizzo

Nel flusso del testo

Perchè una mappa sensibile client-side funzioni, è necessario includere da qualche parte nel documento un set di coordinate e gli URI che definiscono le regioni che saranno sensibili al mouse (un set e un URI per ciascuna regione che può essere cliccata o selezionata dall'utente).

Bisogna includere tali coordinate e link come valori di attributi all'interno di un elemento **`a`** o **`area`**; l'insieme degli elementi **`a`** o **`area`** vanno specificati tra il tag **`<map>`** e il suo tag di chiusura **`</map>`**.

Il valore dell'attributo **`name`** nell'elemento **`map`** deve corrispondere al nome dell'attributo **`usemap`** usato nell'elemento **`img`**. Il nome deve essere univoco nel documento e non utilizzato da altri elementi **`map`**, ma più di una mappa sensibile può far riferimento alle stesse specifiche **`map`**.

8.9 L'elemento area

<area>

Funzione

Definisce coordinate e link per una regione su una mappa sensibile "client-side"

Attributi

`alt, coords, href, nohref, shape`

Contiene

Nessun contenuto

Utilizzo

Nel contesto dell'elemento **map**

Il cuore di una mappa sensibile "client-side" sono gli elementi **<area>**, ciascuno dei quali definisce la regione dell'immagine sensibile al mouse e la relativa azione da prendere da parte del browser.

La regione definita in un elemento **area** agisce esattamente come ogni altro hyperlink: quando il mouse passa sulla zona sensibile, il puntatore del mouse cambia aspetto (tipicamente in una piccola mano) e il browser può mostrare l'URI del collegamento relativo nella barra di stato.

L'attributo **alt**, come il corrispondente dell'elemento **img**, inserisce un testo alternativo che i browser recenti evidenziano al passaggio del mouse. I browser non grafici possono utilizzare il testo alternativo per presentare la mappa sensibile come una lista di collegamenti ipertestuali.

8.9.1 L'attributo shape

L'attributo **shape** serve a definire la forma della regione della mappa sensibile;
i valori possibili sono:

- **circ** o **circle** (per un cerchio);
- **poly** o **polygon** (per un poligono);
- **rect** or **rectangle** (per un rettangolo).

Il valore di **shape** determina come il browser interpreta l'attributo **coords**; se l'attributo viene omesso, viene assunto il valore default.

LaTeX-impaginato

In ottemperanza agli standard, default significa che l'area ricopre l'intera immagine. Se non si specifica un attributo **shape** e non si includono quattro coordinate con l'elemento, il browser ignora del tutto l'area sensibile.

Alcune versioni datate di browser non riconoscono la versione intera dei nomi degli attributi (ad esempio "**rectangle**") e si raccomanda quindi di utilizzare la versione abbreviata (ad esempio "**rect**").

8.9.2 L'attributo coords

- L'attributo, obbligatorio, definisce le coordinate di una regione sensibile al mouse;
- il numero delle coordinate e il loro significato dipendono dall'attributo **shape**;
- si possono definire aree sensibili di una immagine come rettangoli, cerchi o poligoni;
- se le coordinate di un elemento **area** si sovrappongono a quelle di un altro elemento **area**, prendono la precedenza quelle del primo elemento specificato;
- per rettangoli, **coords="x1,y1,x2,y2 "**;
- per cerchi, **coords="x,y,r "**;
- per poligoni, **coords="x1,y1,x2,y2,x3,y3,..."**.

L'attributo obbligatorio **coords** dell'elemento **area** definisce le coordinate di una regione sensibile al mouse in una mappa sensibile client-side. Il numero delle coordinate ed il loro significato dipendono dalla forma della regione sensibile, come definita nell'attributo **shape**.

Il valore opportuno per ciascuna forma include:

circle o **circ**

> **coords="x,y,r"**, dove **x** e **y** definiscono la posizione del centro del cerchio e `r` rappresenta il raggio in pixel.

polygon o **poly**

> **coords="x1,y1,x2,y2,x3,y3,..."**, dove ciascuna coppia di coordinate **x** e **y** definiscono i vertici del poligono. Devono essere definiti almeno tre vertici, ed il poligono viene chiuso automaticamente, senza bisogno di ripetere la prima coppia di coordinate alla fine della lista.

rectangle o **rect**

> **coords="x1,y1,x2,y2"**, dove la prima coppia di coordinate rappresenta un vertice del rettangolo e la seconda coppia il vertice opposto.

Se le coordinate di un elemento **area** si sovrappongono a quelle di un altro elemento **area**, prendono la precedenza quelle del primo elemento specificato.

L'attributo **href** dell'elemento **area** definisce l'URI del collegamento desiderato quando la regione nella mappa sensibile viene cliccata.

Esempio di mappa sensibile

Questo frammento di codice HTML fa visualizzare una cartina delle regioni italiane con la regione Toscana come area sensibile. Al passaggio del mouse (evento **onmouseover**) la Toscana viene evidenziata in colore e al clic viene aperta una nuova finestra (evento javascript:**window.open**) con una immagine della Toscana più dettagliata.

```
<img name="italia0" src="images/italia.gif" width="308"
height="377" usemap="#italia" alt="" />
<map name="italia"> <area shape="poly"
coords="80,108,82,106,80,105,79,102,76,101,74,98,76,95,80,95,83,97,85,9
8,87,99,90,101,
94,104,95,105,98,107,101,108,104,109,106,109,111,109,112,106,114,104,11
6,105,119,109,122,109,123,114,127,
117,130,119,133,121,135,122,137,124,138,123,141,123,140,125,139,129,137
,129,133,132,133,136,135,137,135,
140,132,140,128,145,130,147,130,153,129,155,124,158,123,160,124,161,123
,164,121,165,121,169,116,171,115,
173,112,171,110,173,107,172,108,170,109,168,110,166,107,164,104,160,100
,157,99,153,97,151,93,152,91,155,
91,159,90,158,87,158,86,159,85,158,84,157,84,155,87,155,89,154,92,152,9
3,151,94,146,93,145,94,144,94,139,
91,136,90,134,88,132,87,124,86,119,86,114,83,110,80,109"
href="#"
onclick="javascript:window.open('toscana.html','wintos','width=270,heig
ht=317')" onmouseover="if (document.images)
document.italia0.src='images/italia tosc.gif';" onmouseout="if
(document.images) document.italia0.src='images/italia.gif';">
</map>
```

Massa Carrara
Pistoia
Lucca
Prato
Pisa
FIRENZE
Livorno
Arezzo
Siena
Grosseto

8.10 Le immagini di sfondo

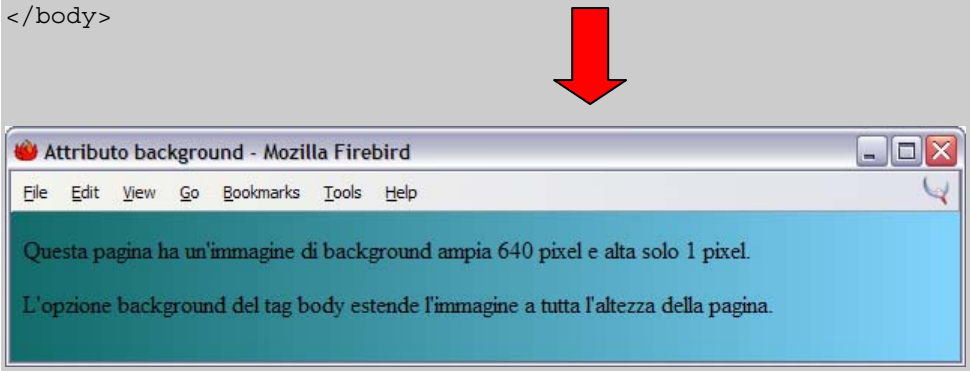
Un'altra tipologia di immagine è qulla che viene usata pre definire uno sfondo per la pagina html. Per default il browser, se l'immagine è più piccola della pagina attuale, ripete lo sfondo nelle due direzioni fino a riempire completamente la pagina;

Per ottonere uno sfondo per la propria pagina HTML si utilizza la proprietà **background** applicata all'elemento **body** .

Esempio:

Questo esempio di codice mostra una pagina con uno sfondo colorato sfumato.

```
<body style="background:url('images/gradiente.png')">
  <p>Questa pagina ha un'immagine di background ampia 640 pixel e
     alta solo 1 pixel.</p>
  <p>L'opzione background dell'elemento body estende l'immagine a
     tutta
     l'altezza della pagina.</p>
</body>
```



Le specifiche CSS forniscono una serie di proprietà che applicate all'elemento **body** permettono di specificare i colori per il testo, per i link e per il colore di sfondo della pagina.

8.11 I colori del documento

Con i browser più popolari e seguendo le specifiche dei Cascading Style Sheets (CSS), è possibile specificare il colore a schermo per i vari elementi del documento.

Tale specifica del colore avviene mediante un valore o mediante un nome.

8.11.1 I valori dei colori

Il valore del colore per gli elementi HTML viene espresso con un numero esadecimale di 6 digit che rappresenta le componenti di rosso, verde e blu (RGB) del colore. Più precisamente i primi due digit indicano la componente di rosso, il terzo e quarto quella del verde, gli ultimi due quella del blu.

Il valore esadecimale 00 (decimale 0) corrisponde al componente di colore completamente assente; il valore esadecimale FF (decimale 255) al componente di colore pienamente saturo.

Il rosso pieno sarà quindi FF0000, il verde 00FF00 e il blu 0000FF, mentre gli altri colori primari derivano da combinazioni di due componenti di colore: il giallo (FFFF00), il magenta (FF00FF) e il ciano (00FFFF): agli estremi il bianco (FFFFFF) e il nero (000000).

Per definire il colore di un elemento, si usa il cancelletto (#) più la tripletta dei colori RGB.

Esempio:

La porzione di codice che segue imposta a magenta il colore dei link visitati:

```
  <style type="text/css">
    a:visited {color:#FF00FF;}
  </style>
...
  <body>
  ...
  </body>
```

8.11.2 I nomi dei colori

Per rendere più semplice l'utilizzo dei colori, essi possono essere richiamati anche tramite un nome. Negli standard sono definiti 16 nomi, che sono validi e riconosciuti in tutti i posti dove è possibile usare un valore di colore.

I nomi di questi 16 colori di base (e il corrispondete valore RGB) sono:

aqua (#00FFFF), gray (#808080), navy (#000080), silver (#C0C0C0), black (#000000), green (#008000), olive (#808000), teal (#008080), blue (#0000FF), lime (#00FF00), purple (#800080), yellow (#FFFF00), fuchsia (#FF00FF), maroon (#800000), red (#FF0000), white (#FFFFFF).

I browser più popolari e più recenti vanno oltre gli standard e supportano parecchie centinaia di nomi di colori (quelli definiti per l'utilizzo col sistema grafico X Window).

8.11.3 La mappa standard

Molti utenti ancora oggi possono visualizzare solo 256 colori, a causa delle limitazioni della scheda grafica del proprio computer; in tal caso se il browser deve visualizzare un colore della pagina HTML che non combacia con uno dei suoi 256, o utilizza quello più simile o fa un'operazione di "dithering", mescolando due o più colori per ottenerne quello richiesto.

È possibile evitare questa operazione (oltretutto in qualche modo dispendiosa in termini di visualizzazione) utilizzando una mappa "standard" di 216 colori al momento della creazione dell'immagine.

Nella mappa standard esistono sei varianti di rosso, di verde e di blu, che combinate in tutti i possibili modi danno questi 216 colori (6 x 6 x 6). Queste varianti hanno valori decimali di saturazione di 0, 51, 102, 153, 204, e 255, corrispondenti a valori esadecimali di 00, 33, 66, 99, CC e FF.

Colori tipo #003333 (dark cyan) o #999999 (medium gray) esistono direttamente nella mappa standard e non subiranno conversioni da parte del browser.

CSSm-pagina2

Questa è la rappresentazione della mappa standard (*safe palette*)

000000 R: 000 G: 000 B: 000	000033 R: 000 G: 000 B: 051	000066 R: 000 G: 000 B: 102	000099 R: 000 G: 000 B: 153	0000CC R: 000 G: 000 B: 204	0000FF R: 000 G: 000 B: 255	003300 R: 000 G: 051 B: 000	003333 R: 000 G: 051 B: 051	003366 R: 000 G: 051 B: 102	003399 R: 000 G: 051 B: 153	0033CC R: 000 G: 051 B: 204	0033FF R: 000 G: 051 B: 255	006600 R: 000 G: 102 B: 000	006633 R: 000 G: 102 B: 051	006666 R: 000 G: 102 B: 102	006699 R: 000 G: 102 B: 153	0066CC R: 000 G: 102 B: 204	0066FF R: 000 G: 102 B: 255
009900 R: 000 G: 153 B: 000	009933 R: 000 G: 153 B: 051	009966 R: 000 G: 153 B: 102	009999 R: 000 G: 153 B: 153	0099CC R: 000 G: 153 B: 204	0099FF R: 000 G: 153 B: 255	00CC00 R: 000 G: 204 B: 000	00CC33 R: 000 G: 204 B: 051	00CC66 R: 000 G: 204 B: 102	00CC99 R: 000 G: 204 B: 153	00CCCC R: 000 G: 204 B: 204	00CCFF R: 000 G: 204 B: 255	00FF00 R: 000 G: 255 B: 000	00FF33 R: 000 G: 255 B: 051	00FF66 R: 000 G: 255 B: 102	00FF99 R: 000 G: 255 B: 153	00FFCC R: 000 G: 255 B: 204	00FFFF R: 000 G: 255 B: 255
330000 R: 051 G: 000 B: 000	330033 R: 051 G: 000 B: 051	330066 R: 051 G: 000 B: 102	330099 R: 051 G: 000 B: 153	3300CC R: 051 G: 000 B: 204	3300FF R: 051 G: 000 B: 255	333300 R: 051 G: 051 B: 000	333333 R: 051 G: 051 B: 051	333366 R: 051 G: 051 B: 102	333399 R: 051 G: 051 B: 153	3333CC R: 051 G: 051 B: 204	3333FF R: 051 G: 051 B: 255	336600 R: 051 G: 102 B: 000	336633 R: 051 G: 102 B: 051	336666 R: 051 G: 102 B: 102	336699 R: 051 G: 102 B: 153	3366CC R: 051 G: 102 B: 204	3366FF R: 051 G: 102 B: 255
339900 R: 051 G: 153 B: 000	339933 R: 051 G: 153 B: 051	339966 R: 051 G: 153 B: 102	339999 R: 051 G: 153 B: 153	3399CC R: 051 G: 153 B: 204	3399FF R: 051 G: 153 B: 255	33CC00 R: 051 G: 204 B: 000	33CC33 R: 051 G: 204 B: 051	33CC66 R: 051 G: 204 B: 102	33CC99 R: 051 G: 204 B: 153	33CCCC R: 051 G: 204 B: 204	33CCFF R: 051 G: 204 B: 255	33FF00 R: 051 G: 255 B: 000	33FF33 R: 051 G: 255 B: 051	33FF66 R: 051 G: 255 B: 102	33FF99 R: 051 G: 255 B: 153	33FFCC R: 051 G: 255 B: 204	33FFFF R: 051 G: 255 B: 255
660000 R: 102 G: 000 B: 000	660033 R: 102 G: 000 B: 051	660066 R: 102 G: 000 B: 102	660099 R: 102 G: 000 B: 153	6600CC R: 102 G: 000 B: 204	6600FF R: 102 G: 000 B: 255	663300 R: 102 G: 051 B: 000	663333 R: 102 G: 051 B: 051	663366 R: 102 G: 051 B: 102	663399 R: 102 G: 051 B: 153	6633CC R: 102 G: 051 B: 204	6633FF R: 102 G: 051 B: 255	666600 R: 102 G: 102 B: 000	666633 R: 102 G: 102 B: 051	666666 R: 102 G: 102 B: 102	666699 R: 102 G: 102 B: 153	6666CC R: 102 G: 102 B: 204	6666FF R: 102 G: 102 B: 255
669900 R: 102 G: 153 B: 000	669933 R: 102 G: 153 B: 051	669966 R: 102 G: 153 B: 102	669999 R: 102 G: 153 B: 153	6699CC R: 102 G: 153 B: 204	6699FF R: 102 G: 153 B: 255	66CC00 R: 102 G: 204 B: 000	66CC33 R: 102 G: 204 B: 051	66CC66 R: 102 G: 204 B: 102	66CC99 R: 102 G: 204 B: 153	66CCCC R: 102 G: 204 B: 204	66CCFF R: 102 G: 204 B: 255	66FF00 R: 102 G: 255 B: 000	66FF33 R: 102 G: 255 B: 051	66FF66 R: 102 G: 255 B: 102	66FF99 R: 102 G: 255 B: 153	66FFCC R: 102 G: 255 B: 204	66FFFF R: 102 G: 255 B: 255
990000 R: 153 G: 000 B: 000	990033 R: 153 G: 000 B: 051	990066 R: 153 G: 000 B: 102	990099 R: 153 G: 000 B: 153	9900CC R: 153 G: 000 B: 204	9900FF R: 153 G: 000 B: 255	993300 R: 153 G: 051 B: 000	993333 R: 153 G: 051 B: 051	993366 R: 153 G: 051 B: 102	993399 R: 153 G: 051 B: 153	9933CC R: 153 G: 051 B: 204	9933FF R: 153 G: 051 B: 255	996600 R: 153 G: 102 B: 000	996633 R: 153 G: 102 B: 051	996666 R: 153 G: 102 B: 102	996699 R: 153 G: 102 B: 153	9966CC R: 153 G: 102 B: 204	9966FF R: 153 G: 102 B: 255
999900 R: 153 G: 153 B: 000	999933 R: 153 G: 153 B: 051	999966 R: 153 G: 153 B: 102	999999 R: 153 G: 153 B: 153	9999CC R: 153 G: 153 B: 204	9999FF R: 153 G: 153 B: 255	99CC00 R: 153 G: 204 B: 000	99CC33 R: 153 G: 204 B: 051	99CC66 R: 153 G: 204 B: 102	99CC99 R: 153 G: 204 B: 153	99CCCC R: 153 G: 204 B: 204	99CCFF R: 153 G: 204 B: 255	99FF00 R: 153 G: 255 B: 000	99FF33 R: 153 G: 255 B: 051	99FF66 R: 153 G: 255 B: 102	99FF99 R: 153 G: 255 B: 153	99FFCC R: 153 G: 255 B: 204	99FFFF R: 153 G: 255 B: 255
CC0000 R: 204 G: 000 B: 000	CC0033 R: 204 G: 000 B: 051	CC0066 R: 204 G: 000 B: 102	CC0099 R: 204 G: 000 B: 153	CC00CC R: 204 G: 000 B: 204	CC00FF R: 204 G: 000 B: 255	CC3300 R: 204 G: 051 B: 000	CC3333 R: 204 G: 051 B: 051	CC3366 R: 204 G: 051 B: 102	CC3399 R: 204 G: 051 B: 153	CC33CC R: 204 G: 051 B: 204	CC33FF R: 204 G: 051 B: 255	CC6600 R: 204 G: 102 B: 000	CC6633 R: 204 G: 102 B: 051	CC6666 R: 204 G: 102 B: 102	CC6699 R: 204 G: 102 B: 153	CC66CC R: 204 G: 102 B: 204	CC66FF R: 204 G: 102 B: 255
CC9900 R: 204 G: 153 B: 000	CC9933 R: 204 G: 153 B: 051	CC9966 R: 204 G: 153 B: 102	CC9999 R: 204 G: 153 B: 153	CC99CC R: 204 G: 153 B: 204	CC99FF R: 204 G: 153 B: 255	CCCC00 R: 204 G: 204 B: 000	CCCC33 R: 204 G: 204 B: 051	CCCC66 R: 204 G: 204 B: 102	CCCC99 R: 204 G: 204 B: 153	CCCCCC R: 204 G: 204 B: 204	CCCCFF R: 204 G: 204 B: 255	CCFF00 R: 204 G: 255 B: 000	CCFF33 R: 204 G: 255 B: 051	CCFF66 R: 204 G: 255 B: 102	CCFF99 R: 204 G: 255 B: 153	CCFFCC R: 204 G: 255 B: 204	CCFFFF R: 204 G: 255 B: 255
FF0000 R: 255 G: 000 B: 000	FF0033 R: 255 G: 000 B: 051	FF0066 R: 255 G: 000 B: 102	FF0099 R: 255 G: 000 B: 153	FF00CC R: 255 G: 000 B: 204	FF00FF R: 255 G: 000 B: 255	FF3300 R: 255 G: 051 B: 000	FF3333 R: 255 G: 051 B: 051	FF3366 R: 255 G: 051 B: 102	FF3399 R: 255 G: 051 B: 153	FF33CC R: 255 G: 051 B: 204	FF33FF R: 255 G: 051 B: 255	FF6600 R: 255 G: 102 B: 000	FF6633 R: 255 G: 102 B: 051	FF6666 R: 255 G: 102 B: 102	FF6699 R: 255 G: 102 B: 153	FF66CC R: 255 G: 102 B: 204	FF66FF R: 255 G: 102 B: 255
FF9900 R: 255 G: 153 B: 000	FF9933 R: 255 G: 153 B: 051	FF9966 R: 255 G: 153 B: 102	FF9999 R: 255 G: 153 B: 153	FF99CC R: 255 G: 153 B: 204	FF99FF R: 255 G: 153 B: 255	FFCC00 R: 255 G: 204 B: 000	FFCC33 R: 255 G: 204 B: 051	FFCC66 R: 255 G: 204 B: 102	FFCC99 R: 255 G: 204 B: 153	FFCCCC R: 255 G: 204 B: 204	FFCCFF R: 255 G: 204 B: 255	FFFF00 R: 255 G: 255 B: 000	FFFF33 R: 255 G: 255 B: 051	FFFF66 R: 255 G: 255 B: 102	FFFF99 R: 255 G: 255 B: 153	FFFFCC R: 255 G: 255 B: 204	FFFFFF R: 255 G: 255 B: 255

8.12 Gli editor di immagini

I programmi di fotoritocco fanno molto di più di quello che il nome sta ad indicare, e spesso per la creazione o la modifica di immagini su Web potrebbe bastare un sottoinsieme delle funzioni messe a disposizione.

Alcune caratteristiche sono però indispensabili per una facile ma accurata produzione di risultati. Quasi tutte le funzioni di ritocco si applicano su aree definite dell'immagine, ed allora il primo strumento necessario è un buon strumento di selezione (*selection tool*) delle aree su cui applicare le modifiche.Uno strumento di selezione davvero utile dovrebbe comprendere la regolazione della tolleranza, che permette di scegliere la precisione con cui lo strumento deve rispondere ai criteri impostati.

La gestione dei livelli (*layers*) è un'altra caratteristica determinante per un buon software di editing di immagini. È possibile separare in livelli differenti le diverse porzioni che compongono un'immagine, (ad esempio lo sfondo e la parte delle scritte testuali), mantenendo ogni parte dell'immagine composta sotto forma di oggetto separato, modificabile singolarmente.

Le maschere (*masking*) sono più flessibili delle selezioni, perché è possibile usare altri strumenti, come filtri e pennelli, per modificarle. Se, per esempio, si vuole applicare un effetto grafico in modo che una certa parte dell'immagine riceva il trattamento al 100 per cento e un'altra parte soltanto al 25 per cento, si può usare una maschera per definire le modalità di applicazione dell'effetto.

La possibilità di porre delle scritte su un'immagine (*text addition*) può essere comoda se non si vuole utilizzare successivamente un programma di desktop publishing. I migliori software di fotoritocco possono scrivere testi su un percorso curvilineo, dipingere il testo con innumerevoli sfumature di colore e regolare il kerning ed il leading (ovvero la spaziatura verticale e orizzontale tra le lettere).

La trasparenza (*transparency*) è un'altra comoda funzionalità, che consente ai livelli sottostanti di essere visibili attraverso livelli superiori che normalmente li coprirebbero. Per esempio, si può mettere un logo aziendale su una porzione di un'immagine, continuando tuttavia a vedere l'immagine sottostante.

Un altro strumento che può risultare utile è il duplicatore (*cloning*), che può copiare porzioni di un'immagine o rimuovere imperfezioni copiando i pixel "puliti" sopra quelli "sporchi".

Tutte le applicazioni sono in grado di importare ed esportare diversi formati, tra cui TIFF (Tag Image File Format) di alta qualità e JPEG (Joint Photographic Experts Group).

Diversi programmi riescono a importare file in formato PSD (Program Support Document), proprio di Photoshop, ma nessuno riesce a mantenere i livelli e altre caratteristiche contenute in tali file.

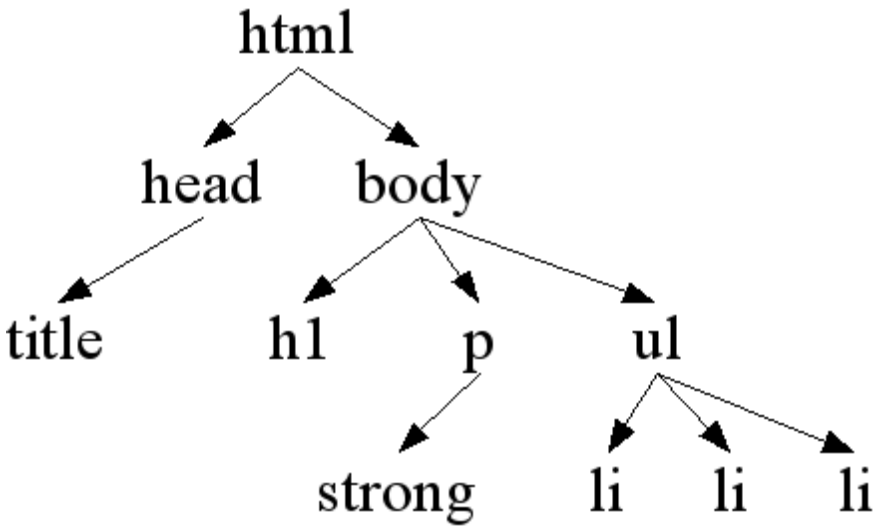
Sono invece pochi i programmi che riescono a importare i file grafici EPS (Encapsulated Postscript) creati da programmi vettoriali come Adobe Illustrator o Macromedia Freehand. In questi programmi, i grafici appaiono nitidi a qualunque livello di ingrandimento. Ma quando si importa un file EPS in un editor di immagini, lo si deve convertire e scegliere la risoluzione a cui lo si vuole usare. Solamente Paint Shop Pro può importare grafici EPS generati da Illustrator.

Infine qualche nome di applicativo per fotoritocco, tenendo presente che è molto più proficuo utilizzare un prodotto di cui si conoscono al meglio le potenzialità che utilizzarne un altro più potente ma che sappiamo sfruttare solo al 50%: Adobe Photoshop, Jasc Paint Shop Pro, Microsoft Picture It!, Ulead Photoimpact.

Struttura ad albero, elementi permessi e deprecati

9.1 Struttura della pagina HTML

Nella figura seguente è rappresentata la struttura ad albero semplificata di una pagina HTML d'esempio.



Gli elementi sono disposti partendo dal padre (`html`) e mettendo sotto ciascuno degli elementi contenuti: i figli. I figli dell'elemento `body` sono gli elementi racchiusi tra i tag di apertura e chiusura dell'elemento `body`, cioè il padre: nell'esempio i figli diretti sono `h1, p e ul`.

9.1.1 Elementi permessi

In un qualsiasi punto della struttura è possibile usare soltanto un sottoinsieme di tutti gli elementi HTML. Il sottoinsieme permesso in un particolare elemento è indicato nella specifica HTML del W3C e queste regole sono descritte schematicamente anche nella Document Type Definition (DTD).

Dette DTD del W3C sono normative, cioè dettano le regole per ciò che riguarda la struttura di una pagina HTML.

Come esempio si veda la DTD dell'XHTML 1.0: XHTML1.0 Strict - DTD http://www.w3.org/TR/xhtml1/dtds.html#a_dtd_XHTML-1.0-Strict

Nella visione XML si preferisce alla DTD il formato **XMLSchema**, che oltre ad avere i vantaggi dell'XML, da' una serie di informazioni aggiuntive non presenti nella DTD. Una riformulazione XML in questo senso delle DTD dell'XHTML 1 la si trova a partire dalla pagina "XHTML1 in XMLSchema http://www.w3.org/TR/xhtml1-schema/".

Ma in pratica dove si può inserire un particolare elemento? Vediamo alcuni esempi significativi:

- non si può inserire un elemento **a** dentro un altro elemento **a**
- non si può inserire una lista (**ol**, **ul**) dentro un paragrafo (**p**)
- non si può specificare più di un **body** o di un **head**

Alcune di queste regole limitano la complessità dell'HTML stesso, altre impongono un senso che altrimenti sarebbe tutt'altro che intuitivo. L'informazione completa delle possibilità (figli e attributi possibili) in ciascun elemento è deducibile dalla DTD. Nell'appendice A si può trovare un riferimento rapido per queste informazioni.

9.1.2 Introduzione alla validazione

Avere pagine HTML valide, con gli elementi correttamente innestati, da' la sicurezza della sua portabilità presente e futura. Per verificare la correttezza delle pagine prodotte si **dovrebbero** usare dei validatori ogni volta che si produce una nuova pagina.

Il W3C mette a disposizione alcuni strumenti per questo scopo che si trovano a partire dalla pagina della Quality Assurance (http://www.w3.org/QA/Tools/#validators).

Il Validatore per HTML/XHTML del W3C (http://validator.w3.org/) è uno dei mezzi con cui si possono verificare le pagine.

9.1.3 Elementi appropriati per il testo

Il corretto uso dei tag per il testo HTML permette di rendere il documento maggiormente leggibile. Di seguito ecco un elenco degli elementi da usare con il corretto significato:

6. **strong** e **em**: per enfatizzare **più** o *meno*
7. **cite**: contiene un riferimento ad un'altra fonte
8. **code**: indica un frammento di codice sorgente di un programma
9. **samp**: indica l'output di un programma, script,...
10. **kbd**: indica del testo che deve essere immesso dall'utente
11. **abbr**: indica la forma abbreviata. Esempi: Eu, Sig.
12. **acronym**: indica un acronimo. Esempi: IP, FIFA
13. **blockquote** e **q** per citare del testo, rispettivamente un blocco di testo o solo poche parole.

9.1.4 Non solo tabelle

Nel web si trovano tabelle usate impropriamente per qualsiasi tipo di dato. Tra gli esempi riconducibili a elementi simili ad un elenco ci sono:

- elenchi con descrizione (materiale, codici, ...);
- elenchi numerati;
- elenchi semplici.

Per dare una struttura a questo tipo di elenchi è bene usare i giusti tag. Ecco un esempio per gli elementi `dl`, `dd`, `dt`, un dizionario:

Uso comune di Definition List

```
<dl>
 <dt>RFC 2822</dt>
 <dd>Internet Message
Format</dd>
 ...
</dl>
```

RFC 2822
Internet Message Format
...

Ecco invece un esempio d'uso di liste numerate:

- Errato

```
<table>
 <tr><td>1</td>
 <td>pippo</td></tr>
 <tr><td>2</td>
 <td>pluto</td></tr>
 <tr><td>3</td>
 <td>paperino</td></tr>
</table>
```

1 pippo
2 pluto
3 paperino

- Corretto

```
<ol>
 <li>pippo
 </li>
 <li>pluto
 </li>
 <li>paperino
 </li>
</ol>
```

- pippo
- pluto
- paperino

9.1.5 Formule su Web

Sul web si trovano facilmente formule inserite come immagini che difficilmente sono correttamente interpretabili a risoluzioni molto diverse da quella per cui sono state realizzate, cioè in pratica non sono ridimensionabili (molto spesso per altro sono realizzato con immagini di bassa qualità).

Per molte formule però, quando sono richiesti l'uso di simboli matematici, si possono usare particolari simboli speciali (entità) e elementi appositi.

Gli elementi **sub** e **sup** racchiudono rispettivamente il pedice e l'apice.

Esempi di formule:

```
&alpha;<sup>2</sup> +
&beta;<sup>2</sup>
= &gamma;<sup>2</sup>
```

$\alpha^2 + \beta^2 = \gamma^2$

```
E = m &middot; c<sup<2</sup>
```

$E = m \cdot c^2$

```
C<sub>3</sub>H<sub>5</sub>
(NO<sub>3</sub>)<sub>3</sub>
&rarr; 12CO<sub>2</sub>
+ 10H<sub>2</sub>O
+ 6N<sub>2</sub>
+ O<sub>2</sub>
```

$C_3H_5(NO_3)_3 \rightarrow 12CO_2 + 10H_2O + 6N_2 + O_2$

Per formule più complesse e altri aspetti attinenti questa problematica è stato introdotto il MathML, che indicare un futuro di integrazione con altre tecnologie XML. Per maggiori informazioni si veda la pagina MathML sul W3C (http://www.w3.org/Math/)

9.1.6 Elementi deprecati

Sul web si fa ancora largo uso di elementi deprecati, per cui sono possibili diverse alternative. Eccone alcuni esempi, con relativa sostituzione XHTML:

- gli elementi **b** e **i** non sono più permessi, al loro posto, anche se sono sensibilmente diversi, è possibile usare rispettivamente **strong** ed **em**.
- L'elemento **font** **non** è permesso nelle nuove specifiche; per cambiare il tipo di carattere (dimensione, famiglia, ...) si usino i CSS (**font-size**, **font-family**, ...).
- L'elemento **center** **non** è permesso dalle nuove specifiche; per cambiare l'allineamento del testo si usano gli style sheet (**text-align**).

Esempio di sostituzione di un vecchio codice HTML con CSS e XHTML:

```
<center><b><font size=3>
 Elementi deprecati
</font></b></center>
```

```
div.importante {
  font-size: 1.1em;
  text-align: center;
}

<div class="importante">
 <strong>Elementi deprecati</strong>
</div>
```

Progettare un sito: parte II

10.1 Web design 2

"A good web site is always the work of many people." [E. Barrett, D. A. Levinson, S. Lisanti – The MIT Guide to teaching Web Site Design – The MIT Press, 2001]

10.1.1 Il team di sviluppo

Potendo disporre di più personale dedicato alla realizzazione del sito web, sarebbe bene suddividere le diverse professionalità in base alle competenze.

Esistono diversi modi per definire chi fa cosa nel web, e con le nuove tecnologie le professionalità tendono sempre più a specializzarsi. In questo paragrafo vedremo alcuni esempi di suddivisione dei ruoli all'interno di un team di progettazione.

Esempio 1: il team di progettazione[30]

Project manager

Information designer

Responsabile del controllo qualità

Responsabile usabilità

Responsabile produzione, production designer

Copy writer, content manager

Art director, visual designer

Responsabile programmazione, back-end engineering

Amministratore di server

Esempio 2: il team secondo gli ambiti di attività[31]

Analizzare/valutare la comunicazione: l'esperto della comunicazione deve capire quali sono le attività comunicative più adeguate.

Progettare: descrivere l'oggetto in modo dettagliato in modo da comprenderne il significato.

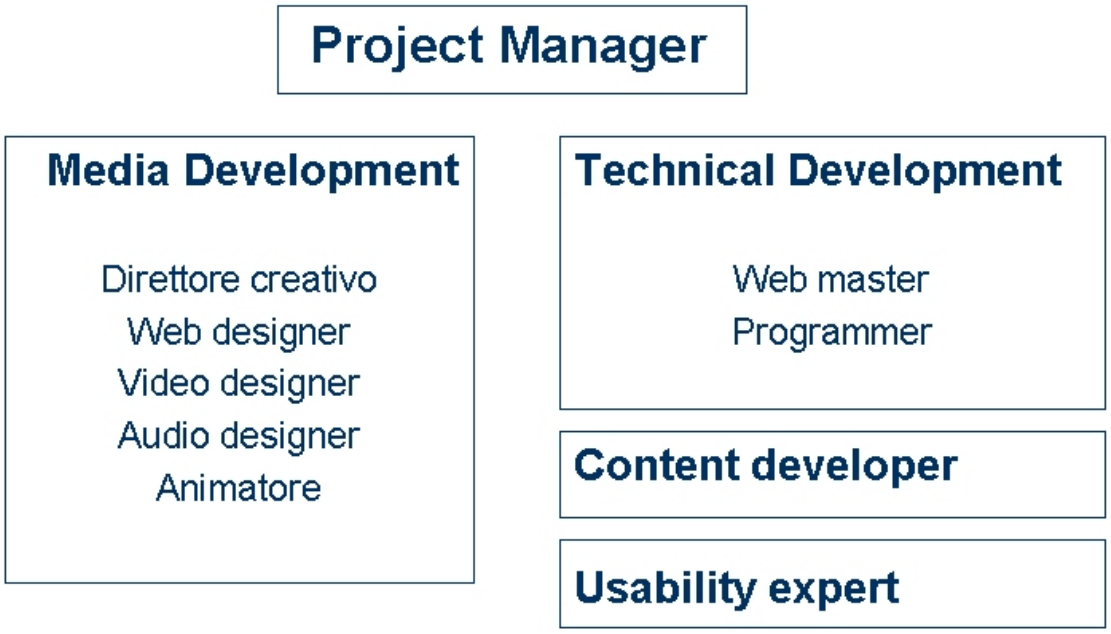
Produrre: inizia solo qui la fase di produzione a cui partecipano gli esperti delle tecnologie, della comunicazione visiva e degli altri codici semiotici coinvolti (audio, video, ecc.).

[30] Goto, Cotler – pg. 78-79

[31] Lorenzo Cantoni e Nicoletta Di Blas – Teoria e tecniche della comunicazione – Apogeo 2002

Mantenere: la comunicazione su Internet deve essere qualcosa di dinamico che richiede manutenzione costante.

Esempio 3: suddivisione per aree[32]



Esempio 4: le nuove figure professionali legati ad Internet[33]

- autore di contenuti (testi, immagini, suoni, ecc.)
- grafico
- information broker (chi naviga in Internet)
- professionista della Comunicazione Pubblica
- programmatore sul Web (HTML, java, altro)
- rilevatore statistico
- traduttore
- venditore di servizi (impaginazione, housing, comunicazione, ecc..)
- webmaster

[32] Cisco Accademy

[33] Forum per la Tecnologia dell'Informazione (FTI) citato in A. Rovinetti – Diritto di parola. Strategie, professioni, tecnologie. Il Sole-24 Ore, Milano 2000

10.1.2 Sviluppare la struttura del sito

Qui la struttura del sito si intende a tre livelli:

- contenuti
- sito
- pagina

che andremo a discutere più in dettaglio di seguito.

10.1.2.1 Contenuti (organizzazione concettuale)

"La suddivisione e la categorizzazione delle pagine è necessariamente determinata dal contenuto[34]*"*. L'organizzazione dei contenuti è una delle prime cose da fare, meglio se il *content management* fa parte dello staff del cliente.

Analisi dell'esistente

Bisogna fare molta attenzione a non fare l'errore di riutilizzare quanto già esiste solo perchè è più semplice da gestire. L'analisi dovrà essere svolta in maniera accurata su tutte le componenti, dal testo alla parte multimediale,, per decidere quello che può rimanere nella ristrutturazione e quello che invece dovrà essere assolutamente eliminato.

Schematizzazione gerarchica

Fare un elenco dei contenuti vecchi e nuovi aiuta senz'altro sia nell'organizzazione generale del web sia a sollecitare quei documenti che arrivano sempre in ritardo, probabilmente anche perchè non vi sono sufficienti risorse dedicate a questo compito. É importante la figura del content manager che gestisca l'intero processo di content delivery.

Pianificare il content delivery

Prevede il dettaglio di quali documenti dovranno essere prodotti o rivisti e quando dovranno essere consegnati. Dovrà essere definito con chiarezza chi fa cosa per quanto riguarda testi, immagini, animazioni o altro).

10.1.2.2 Sito (organizzazione ipertestuale e gerarchica)

"Per fondare la struttura è necessario riuscire ad avere una visione olistica del sito[35]*"*.

Mappatura

[34] op. cit. Goto, Cotler, pag. 25

[35] op. cit. Goto, Cotler, pag. 26

La mappa dovrà visualizzare i link e i percorsi di navigazione. A questo livello non si parla ancora di specifiche tecniche e funzionali. Ogni eventuale anche piccolo cambiamento alla mappa dovrà essere approvato dallo staff.

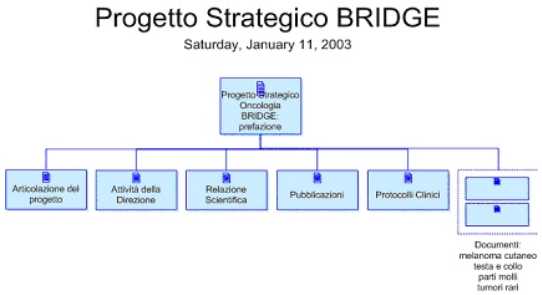


Figure 21 – Esempio di mappa di una parte di sito, create con Microsoft Visio

Analisi dell'esistente

Confrontare la mappa prodotta con quella del sito, se esistente, verificando gli effettivi miglioramenti di navigazione apportati nel redesign. È bene cercare di porsi sempre dal punto di vista dell'utente.

Convenzioni per la nomenclatura

Importante fissare a priori alcune convenzioni sulla nomenclatura dei file in modo da poterli gestire in maniera efficiente e coerente, "*fondamentale sul piano organizzativo*[36]".

Anche la denominazione dei file e altre convenzioni possono aiutare in una migliore organizzazione delle informazioni. Questa è una fase ritenuta "importantissima" in quanto senza una convenzione stabilita i nomi dei file potrebbero essere attribuiti in maniera casuale, disomogenea, incoerente, ma soprattutto "*ogni membro del team deciderà in base alle proprie idiosincrasie o all'umore del momento. A quel punto tirare le fila del progetto e gestire le decine o centinaia di file del sito diventerebbe un'impresa titanica e frustrante*[37]."

Si dovrebbero sempre distinguere due tipi di convenzioni:

- una per la denominazione organizzativa/funzionale (numerica): applicare uno standard numerico aiuta nell'individuare la posizione di un certo contenuto. Considerando la Home Page come 0.0, possiamo nominare le pagine di primo livello come 1.0 … n.0. a loro volta le pagine di secondo livello, per esempio quelle linkabili dalla pagina 1.0, si chiameranno 1.1 … 1.n. In questo modo è facile trovare la posizione dei diversi file, ma non altrettanto conoscere il contenuto delle pagine stesse. Sarebbe quindi auspicabile mantenere due tipi di nominazione, una numerica e una più esplicativa.

[36] op. cit. Goto, Cotler, pag. 27

[37] op. cit. Goto, Cotler, pag. 100

- una per la denominazione HTML: questo è il caso della denominazione più esplicativa rispetto al contenuto del file stesso. Non si hanno certamente delle regole fisse, ma ciascuno può sicuramente usare quelle che preferisce mantenendole però per tutto il sito. Alcuni esempi sul nome di una pagina che parla del profilo dell'azienda potrebbero essere: aboutthecompany.html, about_company.html, co-info.html.

Denominazione organizzativo/funzionale

- **0.0 Home Page**
- **1.0 chi siamo**
  - **1.1. profilo dell'azienda**
  - **1.2 contatti**
  - **….**

Denominazione dei file

- **aboutthecompany.html**
- **about_company.html**
- **co-info.html**

Figure 2 – Esempio di nomenclatura per i file

10.1.2.3 Pagina (organizzazione funzionale e layout di impaginazione)

Gabbia concettuale

La gabbia o schema o content layout, ancora non si preoccupa di colori, immagini o grafici, ma serve per il posizionamento delle informazioni. Deve contenere posizione e dimensione dei testi, delle immagini, dei menu e di ogni altro elemento che si trova nella pagina. Sarebbe bene disegnare una gabbia per ciascun livello di pagina. La costruzione della gabbia facilita il lavoro del visual design e quello di produzione dell'HTML. Il livello di dettaglio che si può avere in una gabbia dipende anche dalle informazioni a nostra disposizione in quel momento.

La gabbia è quindi un layout di informazioni con l'aggiunta di qualche funzionalità e gli item di navigazione. Producendo un foglio di carta per ogni pagina ed affiancandole, è possibile capire meglio il modo con cui l'utente navigherà passando da una pagina all'altra. Questo sistema di testing viene anche denominato prototipo a basso livello e si dimostra sicuramente molto utile nelle prime fasi della progettazione.

Figure 3 – Esempio di gabbia semplice

Riportiamo una lista di elementi che può aiutare nella costruzione di una gabbia:

- immagini/figure/illustrazioni
- contenuti/testi (generici o effettivi, se già disponibili)
- intestazione e/o navigazione globale (barra di navigazione o elementi globali che ricorrono in ogni pagina)
- funzionalità (descrizione degli elementi funzionali/interattivi di base)
- link primari (percorsi di navigazione)
- link secondari
- media (se previsti)
- dimensioni standard della finestra
- meta-informazioni (nell'intestazione e nel piè di pagina della gabbia indicate il nome del progetto, il nome della pagina, il numero della versine, la data, l'autore, il copyright)

Navigazione

Attenzione a non fare l'errore di mantenere la stessa navigazione del sito esistente. Spesso le necessità di un redesign nascono proprio da problemi di navigazione incontrate dagli utenti. Verificare bene se il modello di navigazione esistente può essere corretto o deve essere completamente riorganizzato.

Alcuni modelli di navigazione esistenti sono:

- **navigazione diretta:** l'utente viene indirizzato su scelte predefinite. Spesso viene adottato questo tipo di navigazione per selezionare i contenuti in base all'utente/cliente.
- **navigazione basata su ricerca libera:** è molto spesso il caso dei motori di ricerca o comunque di quei siti con elevata densità di informazioni che quindi permettono all'utente di arrivare direttamente alla pagina che interessa.
- **navigazione per categorie:** le informazioni vengono suddivise in categorie, "adottando la metafora di cartelle e relative etichette/linguette" che sembra risultare più intuitiva per la maggior parte degli utenti in quanto più simile a quella dei sistemi operativi per computer.
- **navigazione lineare:** l'utente viene forzato in un percorso con tasti di avanti e indietro. Non è un sistema comune soprattutto come sistema primario di navigazione: viene spesso usato in sezioni particolari, come un catalogo.

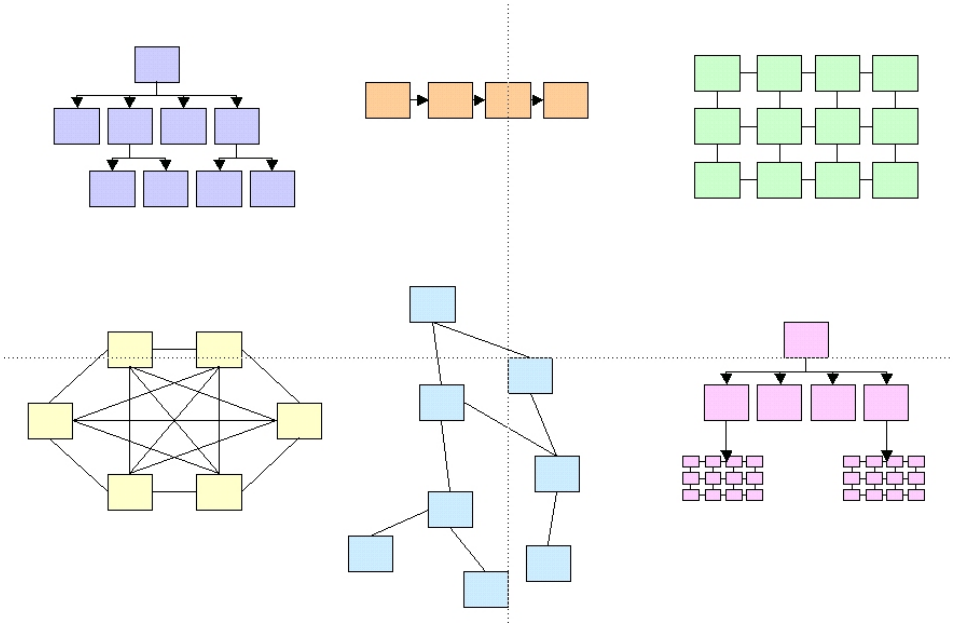


Figure 4 – Esempi di topologia dei siti web, ovvero possibili navigazioni tra le pagine: dall'alto verso destra ad albero o gerarchica, lineare o in sequenza, a matrice o griglia, full mesh, a rete, ibrida

Nomenclatura e titolazioni

Questa è una parte molto importante che definisce il tono della comunicazione del sito. "*I nomi che assegnate ai pulsanti, alle icone e a tutti gli elementi di navigazione dice molto riguardo alla vostra azienda e al vostro approccio. Fate riferimento agli obiettivi del sito: il cliente si aspetta un approccio amichevole e alla mano, oppure improntato a professionalità e credibilità?*[38]"

terminologia	Esempio
basato sull'attività	"comprare un auto"
basato sul compito	"compra un auto"
basato sul ruolo	"venditore di auto"
basato sul soggetto	"auto"
basato sull'organizzazione	"reparto vendite"
misto	"auto", "camion", e "reparto servizi clienti"

Figure 5 – Esempi di nomi di nomi per le voci di menu, a seconda del tipo di terminologia che si vuole adottare

Task e percorsi interattivi

[38] op. cit. Goto, Cotler, pag. 108

Sono le parti interattive, quindi form da riempire, transazioni per i pagamenti online, registrazione con login e password. Verificate bene la loro funzionalità e coerenza.

Ricostruire su carta il sito aiuta a capire il tipo di interazione e le modifiche che potrebbero essere apportate vengono fatte molto velocemente. Esistono due tipi di percorsi che si possono seguire navigando all'interno di un sito:

- un **percorso non-funzionale:** cercare una informazione, come contattare l'azienda, che non dipendono da requisiti tecnologici
- un **percorso funzionale:** transazioni elettroniche, login, ricerche, ecc, che dipendono dalla tecnologia in uso

Simulando una navigazione su carta attraverso il sito possiamo capire velocemente quando ridurre il numero dei passi per raggiungere un determinato task, per esempio una transazione, eliminando quindi click superflui aumentando le probabilità che l'utente porti a termine il suo compito.

10.2 UCD – User Centered Design

La persona che utilizzerà un prodotto dovrà essere in grado di:

- indovinare il da farsi
- capire che cosa sta facendo

Finchè è possibile non ci dovrebbero essere istruzioni o ripetizioni di spiegazioni.

Su Web: aumenta l'offerta di informazioni e di conseguenza aumenta la domanda di attenzione. Ma spesso l'attenzione dell'utente è limitata, e questo si traduce in una alta competizione per cercare di catturare e mantenere l'interesse delle persone che visitano il nostro sito. Fondamantale è catturare e mantenere l'attenzione dell'utente senza però caricarlo di impegni mentali non necessari, pena l'abbandono del sito.

"Ho una regola semplice per individuare il cattivo design. Tutte le volte che trovo indicazioni su come usare qualcosa, si tratta di un oggetto progettato male" [Donald Norman]

Donald Norman è co-fondatore della Nielsen-Norman Group (Jakob Nielsen viene definito il guru dellusabilità), è stato vice-presidente del Gruppo di Tecnologia Avanzata della Apple Computers e membro dellesecutivo di Hewlett Packard e Unext, una compagnia per la formazione a distanza. È professore emerito allUniversità della California, San Diego, dove ha fondato il Dipartimento di Scienze Cognitive ed è stato chair del Dipartimento di Psicologia. È autore di numerosi libri diventati famosi, fra cui La caffettiera del masochista, Il computer invisibile, Le cose che ci fanno intelligenti. Ha scritto un nuovo libro in uscita nel 2004 dal titolo Emotional Design. Why we love (or hate) everyday things. Per maggiori informazioni su Donald Norman, vedere il suo sito al'lindirizzo http://www.jnd.org/.

10.2.1 Progettazione Centrata sull'Utente

1933 → Motto dell'Esposizione Universale di Chicago (Progettazione centrata sulla macchina)

1993 → Motto centrato sull'individuo in vista del ventunesimo secolo (Donald Norman)

1933 Chicago	1993 Norman
La scienza scopre	L'uomo scopre
L'industria applica	La scienza studia
L'uomo si adegua	La tecnologia si adegua

Come dice Norman nella prefazione a "Le cose che ci fanno intelligenti", la nostra società ha involontariamente assunto un orientamento centrato sulle macchine, che antepone le esigenze della tecnologia a quelle delluomo .... Oggi, è luomo ad essere al servizio della tecnologia, ma non dovrebbe essere così. È necessario rovesciare questa prospettiva e trasformarla in un atteggiamento che privilegi la persona. È da qui che dopo sessanta anni Donald Norman scrive il Motto centrato sullindividuo in vista del ventunesimo secolo.

Come è possibile notare, le persone sono passate dall'ultima alla prima posizione nella lista. Progettare principalmente per lutente dovrebbe essere lo scopo principale di qualsiasi progettista. In un periodo in cui la tecnologia è ormai presente nella vita di tutti i giorni, la persona e le sue esigenze devono sempre di più essere messi al centro della progettazione sin dallinizio delle sue fasi.

10.2.2 Principi di design di Norman

Fornire visibilità

In ogni artefatto tutte le parti funzionali devono essere ben visibili così come devono essere visibili le azioni disponibili. Od anche, il numero delle funzioni disponibili non deve superare eccessivamente il numero dei comandi utilizzabili. Viene fatta una distinzione importante fra **artefatti superficiali**, nei quali tutto è visibile, ed **artefatti interni**, dove una parte dell'informazione viene nascosta. Le maniglie sono esempi di artefatti superficiali: è possibile infatti capire "visivamente" dalla loro forma esterna, come dovranno essere usate. Il telefono o il computer, invece, sono esempi di artefatti interni, che nascondono cioè molte informazioni sulle loro funzioni.

Esempio:
il video registratore, il telefono d'ufficio, ecc..
su web: troppe voci di menu, disorganizzazione funzionale, ecc..

Fornire un buon mapping

Per mapping si intende la corrispondenza fra l'azione e l'intenzione.

I nostri comportamenti sono dettati da due tipi di conoscenza che gli psicologici definiscono, rispettivamente, **conoscenza dichiarativa** (o conoscenza *di*) e **conoscenza procedurale** (o conoscenza del *come*)[39]. Mentre la prima si preoccupa soprattutto della conoscenza di fatti e regole quali "se il semaforo è rosso, mi devo fermare", e quindi facile da elencare per iscritto, il secondo tipo di conoscenza presuppone una pratica ed una dimostrazione ed è pressocché difficile da scrivere o insegnare. Per aumentare e facilitare la conoscenza proveniente dal mondo esterno, spesso i progettisti forniscono diverse informazioni mnemoniche.

Un esempio famoso di mapping naturale è quello riguardante le piastre da cucina[40], che riportiamo in fig. X. l'immagine di fig.X è suddivisa in tre parti, una sulla sinistra, una al centro e una sulla destra e ciascuna parte raffigura tre modalità di rappresentazione della corrispondenza tra fuochi e manopole in una piastra da cucina.

La prima parte a sinistra, definita arbitraria, necessita di una spiegazione dettagliata sulla corrispondenza fra monopola e fuoco che viene acceso. In questo caso infatti, senza una descrizione scritta, ci sarebbero ben 24 combinazioni possibili e molta memoria, per poter accendere uno dei fornelli.

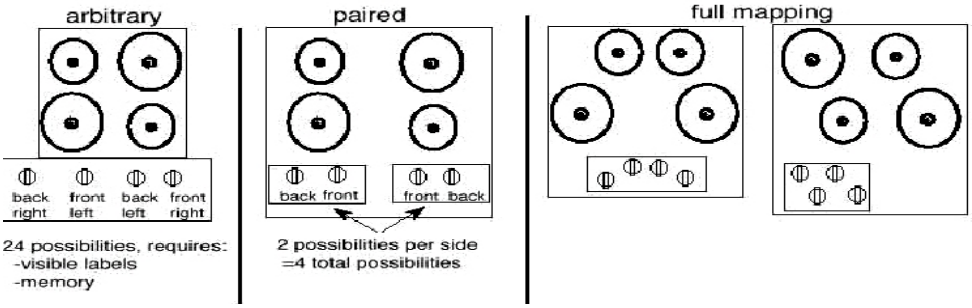


Figure 6 – Mapping naturale dei fornelli della cucina

Il secondo caso, quello centrale, viene definito come disposizione accoppiata. Qui vi sono meno combinazioni possibili, solo 4 in tutto, dovuto proprio ad un piccolo accorgimento: la distanza che separa le coppie di manopole, che visivamente suggerisce la coppia di fuochi che viene comandata. Anche in questo caso, e nonostante siano minori, vi sono comunque delle ambiguità, che possono essere eliminate solamente ponendo una etichetta ben esplicita sotto a ciascuna manopola.

L'ultimo caso, quello a destra nell'immagine, illustra due situazioni possibili in cui è stata eliminata qualsiasi forma di ambiguità. Come vediamo immediatamente, infatti, grazie ad una corrispondenza visiva molto chiara, o mapping naturale, non vi è più alcun dubbio sulla relazione fra manopola e

[39] La caffettiera, pg. 69

[40] La caffettiera, pg. 86

fuoco acceso. Notare che in questi due ultimi casi non si è dovuto ricorrere a nessuna spiegazione, aiuto o esplicita etichetta.

Per concludere, nella progettazione di un prodotto, è bene che le relazioni logico-spaziali fra i comandi, il loro azionamento e il risultato che ne deriva siano il più chiare possibili. Per far ciò conviene sfruttare al meglio le analogie fisiche e i modelli culturali.

esempio:

su web: alcune convenzioni sono già state assimilate dagli utenti, (carrello della spesa nei siti di e-commerce)

Fornire inviti e vincoli all'uso

Ciascun prodotto ha, o dovrebbe avere, delle proprietà e delle funzioni che invitano e vincolano l'utente a fare certe azioni piuttosto che altre.

Gli inviti (o **affordance**) sono infatti quelle proprietà reali e percepite di un oggetto che invitano ad una certa modalità d'uso dello stesso oggetto rendendola chiaramente percepibile. Le forbici per esempio invitano ad essere impugnate in un modo particolare, inserendo il pollice nell'anello più piccolo e più dita contemporaneamente nell'anello più grande. Altri esempi di inviti potrebbero essere le dimensioni di un bottone in una interfaccia grafica: più è grande e più vi sarà la probabilità che verrà notato e usato.

I vincoli (o **constraint**) vengono anche dette funzioni obbliganti proprio perchè vincolano l'uso di un particolare strumento. Esempi di vincoli potrbbero essere alcuni tipi di porte o gli erogatori dei diversi tipi di carburante. In entrambi I casi vi sono costrinzioni all'uso corretto: non è possibile aprire la porta in certe condizioni, l'erogatore di un tipo non entra nei serbatoi di un tipo di carburante diverso. In maniera semplice e naturale queste funzionalità aiutano l'utente a non commettere errori.

Esistono comunque vincoli di diverso tipo:

- **vincoli fisici**: sono limitazioni fisiche che riducono il numero di operazioni possibili. Sono più utili se sono immediatamente visibili e non invitano quindi ad un'azione scorretta.
- **vincoli semantici**: riguardano il significato della situazione per circoscrivere il numero di azioni possibili. I vincoli semantici si basano infatti sulla conoscenza del mondo e della situazione.
- **vincoli culturali**: sono quelli dettati e trasmessi da ogni società. *"Ogni cultura ha un insieme di azioni permesse nelle situazioni sociali. [...] Problemi culturali sono alla base di molti dei problemi che abbiamo con le nuove macchine: non esistono ancora convenzioni o usi consolidati sul modo di trattarle."* E, continua Norman, *"Quelli di noi che studiano queste cose ritengono che le direttive del comportamento culturale siano rappresentate nella mente per mezzo di schemi, strutture cognitive che contengono le regole generali e le informazioni necessarie per interpretare le situazioni e orientare il nostro comportamento. [...] Pericoli più o*

meno gravi aspettano chi viola deliberatamente le cornici convenzionali di una cultura."[41]

- **vincoli logici**: il mapping naturale è un esempio di vincolo logico. Non entrano in gioco fattori fisici o culturali, ma solamente un rapporto logico tra la disposizione spaziale o funzionale dei componenti e le cose del sistema che questi controllano. I vincoli logici non vengono sfruttati quanto dovrebbero, creando quindi ambiguità nella correlazione e confusione nel comportamento.

Esempio:
un paio di forbici invitano ...
la chiave di una porta con la sua serratura constringono ...
su web: link evidenti invitano a cliccare, ...

Fornire feedback

Chiunque interagisca con un sistema dovrebbe sapere in qualsiasi momento, e in modo chiaro, cosa sta succedendo e soprattutto se la sua azione ha avuto l'effetto desiderato. A seconda del sistema vi possono essere feedback di tipo diverso che vanno da un segnale visivo, acustico e a volte anche olfattivo. Il feedback aiuta ad avere sotto controllo ogni azione eseguita e di conseguenza ogni effetto sul sistema. Questo facilita anche la prevenzione di errori soprattutto nei casi in cui vi sono azioni che devono essere eseguite frequentemente e in maniera automatica.

In generale un buon design fornisce informazioni chiare in risposta all'azione dell'utente:

- ciò che l'utente ha fatto è stato recepito dal sistema?

Esempio:
spie di segnalazione sul quadro dell'automobile, i fari sono stati accesi?
su web: dare informazioni di ritorno a seguito di un'azione

Fornire un buon modello concettuale

Un modello concettuale è un'idea mentale che una persona si è fatta sugli altri e sulle cose che li circondano. Ciascuno di noi ha sempre bisogno di darsi delle spiegazioni quando interagisce con qualsiasi elemento dell'ambiente. In ogni caso ciascuno di noi ha anche bisogno odi semplificare l'idea di quello che ci sta intorno per capire meglio il suo funzionamento. Ci vuole del tempo per costruirsi un modello concettuale. Più è buono il modello che ci siamo costruiti, più è facile prevedere gli effetti delle nostre azioni.

[41] norman la caffettiera, pag. 98-99

Un sistema ben progettato ritorna una immagine di sè tale da ridurre la differenza tra il modello del progettista e il modello che l'utente si è creato. Questo permette al sistema di fornire immediatamente le informazioni necessarie a capire la sua struttura e quindi il suo funzionamento.

Esempio:

la cartina della metropolitana di Londra: vi sono gli elementi essenziali per usare bene il sistema, anche se non sono rispettate le relazioni spaziali fra le stazioni

su web: l'immagine del sistema dovrebbe fornire le informazioni essenziali per capire la struttura ed il funzionamento

10.2.3 Il ciclo della UCD

In una progettazione classica, "a cascata", le fasi che vengono eseguite sequenzialmente sono:

- analisi dei requisiti del prodotto/sistema
- design
- implementazione
- validazione
- rilascio e manutenzione del prodotto/sistema

e gli utenti entrano in gioco solo al termine della fase di design. Inoltre i requisiti identificati per il sistema potrebbero rimanere bloccati anche per molto tempo prima che venga realizzato il prodotto. In questo periodo le esigenze degli utenti potrebbero essere cambiate compromettendo quindi le fasi restanti del produzione. Le ultime tre fasi in ogni caso non riusciranno ad influenzare i requisiti o il design del sistema, proprio per la natira a cascata di questo tipo di progettazione.

Nella progettazione centrata sull'utente, invece, i punti interessati sono quelle elencate di seguito, alcune delle quali prevedono una fase ciclica per arrivare ad una conclusione più ottimale del prodotto.

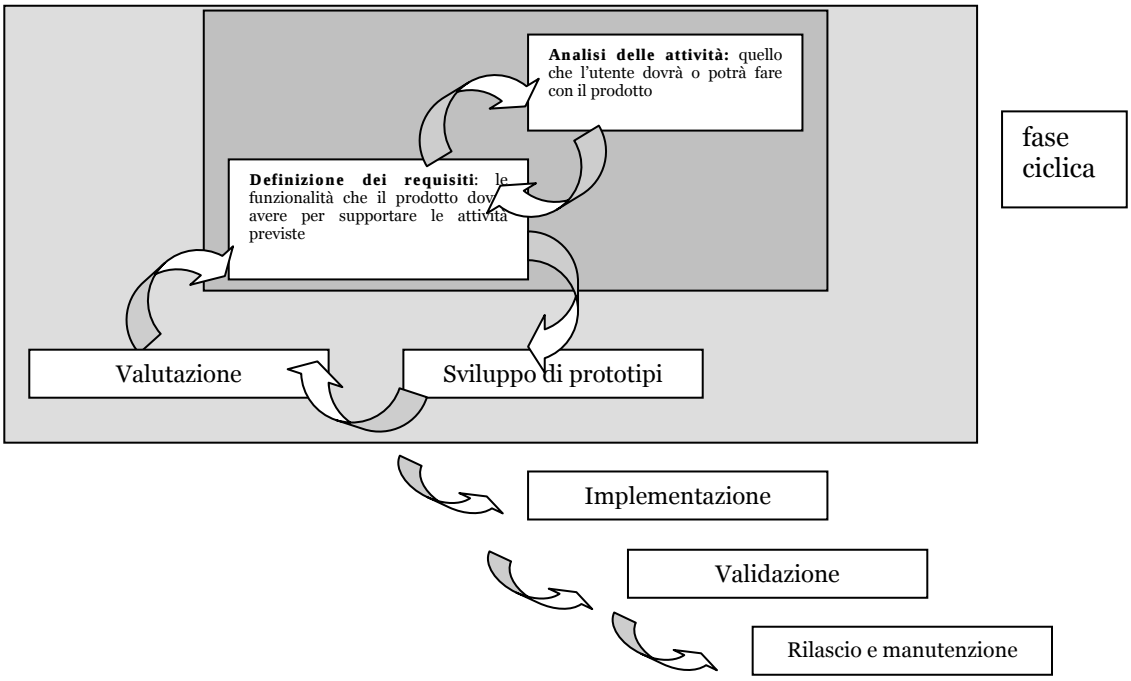


Figure 7 – Ciclo di progettazione della UCD

Come mostrato in figura, la fase ciclica comprende i seguenti passi:

1. **analisi dei requisiti**, suddivisa in due fasi, anch'esse cicliche fino al raggiungimento di un risultato soddisfacente:
   - analisi delle attività che dovranno essere svolte dall'utente attraverso il prodotto/sistema progettato, e
   - definizione dei requisiti che il prodotto/sistema dovrà avere per poter supportare le attività da svolgere
2. **sviluppo di prototipi**, viene fatto solo quando sono stati definiti in modo soddisfacente, i requisiti visti in precedenza
3. **valutazione del prototipo**, con o senza utenti, atti a determinare i requisiti da modificare, quelli da eliminare, quelli da aggiungere

10.3 Usabilità

L'usabilità nasce negli anni '60 nell'ambito dell'ergonomia in relazione a qualunque tipo di interazione uomo-artefatto. Con l'evoluzione informatica l'usabilità ha trovato molti ambiti di applicazione, tra cui ultimamente, il web.

Compito degli studi di usabilità è fare in modo che il modello mentale di chi ha progettato il software (*design model*), da cui deriva il suo reale funzionamento, corrisponda il più possibile al modello mentale del funzionamento del software così come se lo costruisce l'utente finale (*user model*). Questo è possibile tenendo in considerazione le abilità e i limiti fisici dell'uomo, le attività che essi devono svolgere attraverso lo strumento progettato e il contesto in cui lo useranno.

10.3.1 Definizioni di usabilità

Vi sono molte definizioni di usabilità, che in questo contesto elenchiamo per conoscenza:

1. IBM
2. la ISO 9241-11;
3. quella di Jakob Nielsen, considerato il guru dell'usabilità;
4. quella più operativa sviluppata da Rizzo-Marti-Bagnara nel 2001.

IBM

Un qualsiasi artefatto deve essere:

- **utile:** identificando i bisogni degli utenti
- **usabile:** i prodotti devono essere progettati in modo che siano facili da usare
- **desiderabile:** identificare i fattori che motivano gli utenti

ISO 9231/11 - 1992

L'usabilità di un prodotto è determinata dalla sua:

- **efficacia:** utilità, appropriatezza, scopi (sono riuscito a raggiungere il mio scopo?)
- **efficienza:** velocità, affidabilità, scalabilità, numero di errori (quanto mi è costato raggiungere il mio scopo? quante risorse cognitive ho speso? quante volte ho perso inutilmente tempo o sono incappato in errori?)

- **soddisfazione:** come gli utenti si sentono in relazione al sistema: piacevolezza, facilità d'uso, motivazione, interesse, apprendimento (sono soddisfatto dell'interazione avuta con il sistema? è stata un'esperienza positiva o frustrante? tornerò ad interagire con il sistema o sono stufo?)

Jakob Nielsen

Secondo Nielsen usabilità è

"La misura della qualità dell'esperiensa dell'utente in interazione con qualcosa, sia esso un sito web o un'applicazione software tradizionale o qualsiasi altro strumento con il quale l'utente può operare."

E ancora

"Un prodotto è usabile quando è facile da apprendere, consente un'efficienza di utilizzo, è facile da ricordare, permette pochi errori di interazione e di bassa gravità, è piacevole da usare."

Definizione operativa - 2001

"Progettare tecnologie che siano di reale supporto all'attività umana." [A. Rizzo, P. Marti, S. Bagnara, Univ. di Siena – 2001] Ovvero, progettare tecnologie che tengano conto contemporaneamente:

- delle abilità e dei limiti fisici e cognitivi delle persone
- delle attività che essi devono svolgere attraverso lo strumento progettato
- del contesto culturale, fisico ed organizzativo in cui lo useranno

Le motivazioni, le aspettative, il background culturale delle persone, la natura delle attività che esse svolgono, influenzano in modo determinante il modo stesso con cui le informazioni vengono elaborate.

10.3.2 I principali problemi di usabilità

Secondo le ultime statistiche, il 75% dei siti presenta notevoli problemi di usabilità. Il 50% delle transazioni vengono perse per problemi di accesso alle informazioni.

Alcuni problemi importanti di usabilità sono dati dalla ricerca delle informazioni all'interno di un sito o dalla stessa architettura informativa, riconducibili in parte ad una cattiva progettazione della navigabilità.

Navigabilità

Per navigabilità si intende la relazione fra le pagine che compongono un sito web, o comunque un sistema ipermediale. L'architettura informativa del sito determina la navigabilità e aiuta gli utenti a costruirsi i modelli mentali utili per capire l'organizzazione delle informazioni.

La buona navigazione dovrebbe, in ogni momento della visita di un sito web, rispondere alle domande:

- dove sono? in relazione al web, e in relazione alla struttura del sito
- dove sono stato? meglio se viene indicato il percorso fatto dall'utente durante la navigazione (breadcrumbs o briciole di pane)
- dove posso andare ?

Per far questo dobbiamo innanzitutto suddividere i link di un sito web in tre grandi categorie:

- **link topici:** riguardano ulteriori informazioni sull'argomento trattato
- **link strutturali:** mettono in relazione fra loro diverse pagine della struttura del sito, allo stesso livello o a livelli differenti
- **link associativi:** collega altre pagine con contenuti simili o collegati all'argomento trattato

Figure 8 – I diversi tipi di link su una pagina web

Le pagine interne di un sito web dovrebbero tutte avere una intestazione, o navigazione persistente, composta da 5 elementi essenziali:

- ID del sito
- Una via alla HomePage
- Utilities
- Una via per la ricerca
- Le sezioni principali del sito

Questi elementi aiutano a mantenere la coerenza interna del sito, e a rendere il sito più usabile. Una prova importante da poter fare per capire se si sono rispettate le regole appena viste, è rappresentata dal cosiddetto test del portabagagli. Ci deve innanzitutto mettere di fronte ad una qualsiasi pagina interna del sito e provare quindi a rispondere alle seguenti domande:

1) *In che sito mi trovo? ID*
2) *In che pagina sono? Nome della Pagina*
3) *Quali sono le sezioni principali del sito? Sezioni del sito o navigazione primaria*
4) *Quali opzioni ho a disposizione a questo livello? Navigazione locale*
5) *Dove mi trovo nella struttura generale del sito? Indicazioni "voi siete qui"*
6) *Come posso effettuare una ricerca? Campo ricerca*

Se si trovano le risposte, allora stiamo andando verso una buona progettazione.

Per quanto riguarda la HomePage le cose cambiano leggermente. La HomePage infatti è una pagina particolare, molto importante perchè dovrebbe essere la pagina che ci fa entrare nel sito. Per questo motivo può anche non seguire la strutturazione delle pagine interne, e discostarsi leggermente, pur mantenendo gli stessi nomi dei menu. Gli elementi principali che distinguono la HomePage dalle altre pagine sono:

- l'identità del sito, normalmente un logo, deve essere messa ben in evidenza, in modo che chiunque possa subito identificarne la paternità
- mettere in evidenza la mission del sito in modo chiaro e sintetico; per mission si intende una breve frase che spieghi chi siamo, cosa facciamo e i nostri obiettivi
- dovrebbe avere una *tagline*, ovvero poche parole che normalmente accompagnano il logo e aiutano a collegarsi con la mission. Solo le aziende più famose potrebbero non avere la *tagline*.
- può una diversa disposizione delle sezioni e dei link, ma non diversi nomi (consistenza)

10.3.3 Leggi di Krug sull'usabilità

- **Legge prima:** dont' make me think!
- **Legge seconda:** non importa quanti click devo fare, se ogni click è frutto di una scelta che non richiede impegno e che non è ambigua
- **Legge terza:** sbarazzati della metà delle parole di ogni pagina e poi sbarazzati della metà di quello che resta

10.4 Come si misura l'usabilità

Esistono diversi metodi per misurare l'usabilità tra possiamo distinguere:

- Metodi analitici (Questi test non vengono eseguiti su base soggettiva ma su principi stabiliti su base empirica. Sono i più veloci da eseguire in quanto non si ha bisogna di cercare soggetti esterni):
    - analisi dei compiti (task analysis)
    - cognitive walkthrough → aiuta a valutare la facilità d'uso e di apprendimento di una interfaccia secondo gli obiettivi e le aspettative dell'utente e si basa sul Modello dell'Azione di Norman. Può essere svolta solo da esperti o anche con l'aiuto di utenti
    - valutazione euristiche → si basa su euristiche o linee guida e serve a valutare se sono stati rispettati i principi della buona progettazione e viene normalmente eseguita da esperti di usabilità

"La task analysis richiede l'analisi delle componenti del compito, il calcolo dei passi necessari allo svolgimento di una procedura. Nel caso del cognitive walkthrough si tiene anche conto delle caratteristiche cognitive dell'utente (competenze richieste, conoscenze di dominio, eccetera), verificando sull'interfaccia/prototipo l'esistenza di eventuali problemi rispetto alle previsioni.

La valutazione euristica invece valuta l'interfaccia sulla base di liste di euristiche. Tali euristiche sono principi che hanno un elevato valore predittivo perché rappresentano la sintesi dei problemi di usabilità più frequenti organizzati in categorie. Le euristiche di Nielsen, ad esempio, sono ottentute tramite analisi fattoriale su una base di 249 problemi riscontrati in studi di vario tipo."

- Metodi empirici (Sono i test più costosi sia in tempo sia in denaro. Alcuni di questi richiedono uno specifico laboratorio attrezzato con telecamere e altre apparecchiature necessarie al test.):
    - analisi dei tempi di esecuzione
    - questionari di soddisfazione
    - osservazione diretta con annotazione degli errori
    - pensare ad alta voce (thinking aloud)

Non potendo in questo contesto dettagliare tutti i tipi di valutazione dell'usabilità, consideriamo in dettaglio quello più utilizzato, anche perchè prevede costi minori, ovvero la valutazione euristica.

10.4.1 Valutazione euristica

Normalmente la valutazione euristica viene fatta sfruttando euristiche già conosciute e generali, o euristiche specifiche per un contesto o per delle applicazioni particolari:

- linee guida generali (principi di design di Norman, euristiche di Nielsen, ecc..)
- linee guida particolari (uso di colori, icone, etichette, menu, ecc..)
- linee guida per applicazioni specifiche (disabilità, ecc..)

Le 10 euristiche di Nielsen[42] riportate di seguito sono quelle che più in generale tengono in considerazione degli aspetti più importanti di un sito web. Ciascuna euristica a sua volta, per essere soddisfatta dovrà rispondere a specifiche domande o soddisfare certi requisiti. Di seguito riportiamo le euristiche di Nielsen che dovrebbero essere sempre testate quando progettiamo un sito web.

1. **informare l'utente sullo stato del sistema**
    - cosa sta facendo il sistema?
    - che effetti ha avuto o sta avendo l'azione dell'utente?
    - dove si trova l'utente dopo una interazione?
2. **parlare il linguaggio dell'utente**
    - parlare il linguaggio dell'utente
    - usare il modello concettuale dell'utente
    - rispettare le sue aspettative
    - nascondere le informazioni non necessarie o eliminarle
    - raggruppare le informazioni
    - creare delle gerarchie visive delle informazioni (più grandi più importanti ecc.)
3. **dare controllo e libertà di uscita all'utente**
    - fornire sempre delle vie d'uscita chiare e semplici
    - non fornire informazioni di sistema complesse per l'utente
    - link alla home page sempre ben visibile

[42] Jakob Nielsen, 1994 - http://www.useit.com/papers/heuristic/heuristic_list.html

4. essere coerenti e tenere conto degli standard

- coerenza nei comandi (stessi comandi-stessi effetti) e nella loro posizione sulle pagine
- violare uno standard solo se questi sono ben consolidati

5. prevenire gli errori

- funzioni diverse devono essere rappresentate in modo diverso
- fornire messaggi chiari sugli effetti di azioni irreversibili

6. favorire il riconoscimento piuttosto che il ricordo

- le persone ricordano sempre meno
- rendere minimo il carico di memoria dell'utente
- ridurre le informazioni che devono essere tenute a mente
- rendere visibili e chiare le informazioni utili a proseguire nell'interazione

7. fornire delle scorciatoie per gli utenti più esperti

- rendere il sistema flessibile in modo che si adatti al meglio ai diversi livelli di expertise degli utenti

8. adottare un'estetica e un design minimalista

- attenzione alla ridondanza di elementi che possono ridurre la visibilità delle informazioni più rilevanti

9. aiutare l'utente a riconoscere, diagnosticare e recuperare l'errore

- permettere all'utente di ritornare sulle sue azioni
- rendere visibili tutte le azioni che ha svolto
- fornire messaggi di errore chari e con indicazioni sulle possibili vie d'uscita

10. fornire aiuto e documentazione

- fornire documentazione di aiuto solo se strettamente necessaria
- tale documentazione dovrà comunque essere il più possibile chiara, accessibile, centrata sugli scopi e gli obiettivi degli utenti, breve e pratica

10.4.2 test di usabilità

Il test classico di usabilità è fatto direttamente con gli utenti ed è più importante di quanto si possa immaginare, in quanto consente di valutare le reazioni dell'utente a diversi stadi della progettazione. Permette inoltre di poter concentrare il test su alcuni aspetti specifici del prodotto e raffinarli di conseguenza, per poi tornare a svolgere un nuovo test con utenti.

Il test di usabilità infatti può essere fatto anche su un prototipo semi-funzionale del prodotto e non sulla versione definitiva. È dunque sempre meglio condurre un test di usabilità prima della fine del lavoro, per non incappare in aggiustamenti dettati dall'interazioni con gli utenti, che comportino delle modifiche importanti al lavoro già fatto.

Per ulteriori informazioni sulla conduzione dei test di usabilità (anche in remoto) e sull'allestimento di un laboratorio appositamente attrezzato allo scopo, si rimanda ai numerosi testi e siti che si occupano dell'argomento.

CSS: parte II

11.1 CSS predefiniti!

Io non ho specificato nessuno stylesheet ma i miei <h1> vengono in grassetto come mai? Ci sono gli stylesheet predefiniti:

- default stylesheet del browser
- stylesheet delle preferenze utente

Combinando questi stylesheet con quelli indicati nella pagina il browser presenta i contenuti.

In quasi tutti i browser è possibile specificare delle preferenze personali, per esempio:

- Font o famiglie di font da utilizzare per sans-serif, serif, monospace
- Colori per gli iperlink (visitati, da visitare, etc..)
- Dimensione dei font predefinite

Una delle caratteristiche fondamentali dei CSS è che più di uno stylesheet può influenzare la presentazione di un documento. Questa caratteristica è conosciuta come *Cascading (in cascata)* perché i differenti stylesheet vengono eleborati in serie.

Il *Cascading* è una delle caratteristiche fondamentali perché ci permette di influenzare come verrà mostrato il documento attraverso diverse sorgenti:

- il browser
- il web designer
- l'utente

Anche senza l'utilizzo di un CSS probabilmente il vostro browser vi mostrerà le ancore ... di colore blu, ma chi glielo ha detto?

È una delle tante impostazioni predefinite del vostro browser, così come le informazioni su come trattare gli h1, h2, blockquote, etc..

Queste informazioni sono nel *default stylesheet* del vostro browser e vengono unite con quelle degli stylesheet dei documenti e con le preferenze degli utenti per decidere come mostrare il documento.

Il W3C all'indirizzo http://www.w3.org/TR/CSS21/sample.html pubblica un default stylesheet per l'HTML, nato da una ricerca sul comportamento della maggior parte dei vari browser. Il seguente stylesheet è quello che ci aspettiamo come default stylesheet per la maggior parte dei browser:

```
address,
blockquote,
body, dd, div,
dl, dt, fieldset, form,
```

```
frame, frameset,
h1, h2, h3, h4,
h5, h6, noframes,
ol, p, ul, center,
dir, hr, menu, pre   { display: block }
li               { display: list-item }
head             { display: none }
table            { display: table }
tr               { display: table-row }
thead            { display: table-header-group }
tbody            { display: table-row-group }
tfoot            { display: table-footer-group }
col              { display: table-column }
colgroup         { display: table-column-group }
td, th           { display: table-cell; }
caption          { display: table-caption }
th               { font-weight: bolder; text-align: center }
caption          { text-align: center }
body             { margin: 8px; line-height: 1.12 }
h1               { font-size: 2em; margin: .67em 0 }
h2               { font-size: 1.5em; margin: .75em 0 }
h3               { font-size: 1.17em; margin: .83em 0 }
h4, p,
blockquote, ul,
fieldset, form,
ol, dl, dir,
menu             { margin: 1.12em 0 }
h5               { font-size: .83em; margin: 1.5em 0 }
h6               { font-size: .75em; margin: 1.67em 0 }
h1, h2, h3, h4,
h5, h6, b,
strong           { font-weight: bolder }
blockquote       { margin-left: 40px; margin-right: 40px }
i, cite, em,
var, address     { font-style: italic }
pre, tt, code,
kbd, samp        { font-family: monospace }
pre              { white-space: pre }
button, textarea,
input, object,
select           { display:inline-block; }
big              { font-size: 1.17em }
small, sub, sup { font-size: .83em }
sub              { vertical-align: sub }
sup              { vertical-align: super }
thead, tbody,
tfoot            { vertical-align: middle }
td, th           { vertical-align: inherit }
s, strike, del   { text-decoration: line-through }
hr               { border: 1px inset }
ol, ul, dir,
menu, dd         { margin-left: 40px }
ol               { list-style-type: decimal }
ol ul, ul ol,
ul ul, ol ol     { margin-top: 0; margin-bottom: 0 }
u, ins           { text-decoration: underline }
br:before        { content: "\A" }
:before, :after { white-space: pre-line }
center           { text-align: center }
```

```
abbr, acronym   { font-variant: small-caps; letter-spacing: 0.1em
}
:link, :visited { text-decoration: underline }
:focus          { outline: thin dotted invert }
```

11.1.1 CSS multipli

Per ogni documento è possibile specificare più di uno stylesheet.

Gli stylesheet specificati verranno trattati in cascata (da cui *Cascading*).

Se ci sono dichiarazioni uguali in più stylesheet verrà presa in considerazione l'ultima che il browser incontra (leggendo il documento dall'alto verso il basso).

In questo modo si posso combinare vari stylesheet, anche scritti in momenti diversi!

Esempio

```
...
   <link rel="stylesheet" type="text/css" href="base.css" />
   ...
   <style type="text/css">
      body {
        color: blue;
      }
   </style>
   ...
```

11.2 Ancora sui selettori

Nella prima parte del corso abbiamo visto a cosa servono i selettori: *a specificare gli elementi assoggettati alla regola*.

```
selettore
 |
 -
h1  { color : green }
      -------------
            |
      dichiarazione
```

L'esempio più semplice di selettore è quello che individua gli elementi assoggettati tramite il suo *tipo* (h1 nell'esempio).

11.2.1 Le classi e l'attributo CLASS

Per raggruppare le regole di elementi di tipo diverso si creano delle classi di regole. Diversamente dai selettori per tipo le classi permettono così di impostare dichiarazioni su elementi anche di tipo diverso.

Esempio di selettore per classe:

```
    .mia_classe {
  font-weight: bold
}
```

Il `.` (punto) è il delimitatore che indica che il nome successivo rappresenta il nome di una classe.

Ogni elemento HTML può avere associato un attributo `class`.

Esempio di dichiarazione ed utilizzo di una classe

```
<style type="text/css">
  .mia_classe {
  font-weight: bold
}
</style>
```

```
<div class="mia_classe">Testo in grassetto</div>
<p class="mia_classe">Testo in grassetto</p>
```

Le classi sono molto utili per raggruppare regole che riguardano elementi di tipo diverso (nell'esempio soprastante sia il p che il div sono assoggettati alla regola in quanto entrambi di classe “mia_classe”.

11.2.2 Il selettore per ID e l'attributo ID

Ogni elemento HTML può avere associato un attributo `id` attraverso il quale determinare univocamente l'elemento.

Il valore dell'attributo **id** deve essere univoco in tutto il documento.

```
<span id="mio_id">Testo individuato dall'id "mio_id"</span>
```

Per associare una regola dello stylesheet solamente ad un elemento è possibile individuarla tramite il selettore per id.

Il carattere # è il delimitatore che indica che il nome che segue riguarda l'id di un elemento.

Esempio di dichiarazione ed utilizzo del selettore per id

```
<style type="text/css">
  #mio_id { color: blue }
</style>
```

```
<div id="mio_id">Testo individuato dall'id "mio_id"</div>
```

Cosi come il selettore per classe, quello per id permette di assoggettare elementi di tipo diverso.

11.2.3 Combinare i selettori: tipo e classe

I selettori possono però essere combinati per affinare le selezioni individuate.

Combinare classi e tipi è una delle possibilità offerte dai CSS.

Esempio di selettore per tipo e classe

```
<style type="text/css">
  p.importante {
  font-weight: bold
}</style>
```

```
<p class="importante">Paragrafo importante</p>
```

Seleziona tutti gli elementi di tipo (**p**) che hanno class="importante" impostando il font-weight a bold.

Gli elementi di tipo (**p**) che non hanno class="importante" non sono influenzati da questa regola.

Anche il selettore per **id** può essere combinato con gli altri selettori allo stesso modo degli altri selettori.

Esempio di selettore per tipo e id

```
<style type="text/css">
  p#mio_id { color : blue }
</style>
```

```
<p id="mio_id">Paragrafo con id "mio_id"</p>
```

All'interno di un documento HTML l'id deve essere univoco, ma associando lo stesso stylesheet a più documenti è possibile che l'id "mio_id" sia associato ad elementi di tipo diverso. La regola soprastante si applica sono ai **p** con id="mio_id".

11.2.4 Usare più classi

Un elemento HTML può appartenere a più **classi**.

Le classi di appartenenza di un elemento, descritte tramite l'attributo `class` devono essere separate da **spazi** (naturalmente lo spazio non è un carattere ammesso nei nomi di classe).

Esempio di utilizzo di più classi

```
<style type="text/css">
  p.spiegazione {
    color: red
  }
  p.importante {
    font-weight: bold;
  }
</style>

<p class="spiegazione importante">
  Questa è una spiegazione
  importante
</p>
```

Questa è una spiegazione importante.

Nel caso dell'esempio il paragrafo è assoggettato ad entrambe le regole in quanto appartiene ad entrambe le classi, quindi il testo verrà sia rosso che in bold.

11.2.5 Combinare i selettori per contesto

I selettori contestuali prendono in considerazione il contesto in cui gli elementi HTML sono inseriti.

```
selettoreSemplice-1    selettoreSemplice-2    ...    { dichiarazione }
```

- i selettoriSemplici sono selettori del tipo h1, h2, p, p.classe, etc.
- tra un selettoreSemplice ed il successivo devono essere presenti uno o più spazi

Lo spazio tra i selettori indica che gli elementi individuati da selettore-N+1 devono essere discendenti degli elementi individuati dal selettore-N. Un elemento A è discendente di un elemento B quando A è figlio di B oppure A è figlio di uno dei discendenti di B (A è contenuto dentro l'elemento B).

Esempio di utilizzo di selettori per contesto

```
<style type="text/css">
  p span {font-style: italic }
</style>

<p>
  Paragrafo con testo in <span>rosso</span>
</p>
```

Paragrafo con testo in rosso

Individua gli span che sono *discendenti* di un elemento **p**

Gli span con non sono discendenti di un **p** non sono influenzati da questa regola

Esempio di utilizzo di selettori per contesto

```
<style type="text/css">
  div.domanda strong { color: green }
</style>

<div class="domanda">
  div con testo strong
  in <strong>verde</strong>
</p>
```

div con testo strong in **verde**

Individua gli strong che sono figli di un elemento **div** con class="domanda"

Si possono anche effettuare selezioni contestuali più approfondite del tipo:

selettoreSemplice-1 selettoreSemplice-2 selettoreSemplice-3 ... { dichiarazione }

Esempio:

```
  div.libro div.capitolo div.autore { font-weight: bold; color:
blue }
```

Seleziona i div di classe autore che sono discendenti di div di classe capitolo che sono discendenti di div di classe libro

```
div.libro
      |
      -- div.capitolo
               |
               -- div.autore
```

La selezione per contesto permette di specificare altri operatori oltre quello di *discendenza* indicato tramite lo spazio tra due separatori semplici.

Un altro operatore di contesto disponibile nei CSS è il **>**.

Inserendo un **>** si indica che il selettore successivo al > si riferisce ad un figlio dell'elemento che lo precede. **Un figlio, non un discendente!**

Ci sono anche altri operatori di contesto che però non tratteremo, maggiori informazioni sui **selettori** si possono trovare all'indirizzo: http://www.w3.org/TR/CSS21/selector.html

11.2.6 Selettori: *

I CSS prevedono anche un selettore "universale", *.

Esempio

```
body * p { font-weight: bold }
```

Questa regola imposta il font-style a italic a tutti i p che sono figli **non diretti** di body:

deve esserci un qualsiasi elemento tra body e p (* sta per qualsiasi elemento)

Questa dichiarazione non si applica ai figli diretti di body.

```
...
<body>
  <p>A questo paragrafo non si applica la regola</p>
  <span><p>questo paragrafo è invece affetto dalla
dichiarazione</span></p>
...
</body>
```

11.3 Pseudo classi

Le pseudo classi permettono di specificare delle classi "fittizie" per ottenere effetti stilistici particolari.

Esempio di pseudo classe

`a:hover { color: red }`	Link che cambia colore quando ci passiamo sopra (hover)

Le pseudo classi sono classi non contrassegnate da attributi `class`, ma da:

- eventi particolari (come la `hover`)
- da contesti particolari (come la `first-child` che vedremo dopo)
- da proprietà particolari dell'elemento corrente (come la `link` che vedremo dopo).

I `:` (due punti) sono il separatore che indica che la stringa successiva si riferisce ad una pseudo classe.

La pseudo classe deve essere sempre indicata per ultima.

11.3.1 Pseudo Classi: hover, active, focus

Esistono tre pseudo classi disponibili che permettono di cambiare lo stile in funzione di eventi sollevati dall'utente:

- `hover` - quando passo con il mouse sopra l'elemento, quando cioè sono sopra l'elemento
- `active` – quando l'elemento è attivo, tipicamente mentre ci sto cliccando sopra
- `focus` – quando l'elemento è nel focus, che a differenza del hover si può ottenere anche spostandoci con i tab nella pagina

Esempi di pseudi classi

`a:hover { color: red; }`	Testo che cambia colore quando ci passiamo con il mouse
`a:active { color: blue; background-color: pink }`	Testo che cambia sfondo e colore quanto attivato
`a:focus { font-size: 150%; }`	Testo che si ingrandisce quando è nel focus corrente

La pseudo classe `focus` su explorer per windows non funziona.

11.3.2 Link, visited

Esistono anche pseudo classi che verificano "la storia" dell'elemento:

- link (un elemento è di pseudo classe link se è un ancora e non ho mai visitato la destinazione dell'ancora)
- visited (quando l'elemento è un ancora ed il link è già stato visitato)

Esempi di pseudo classi link e visited

```
a:link:hover {
  background-color: yellow;
}
a:visited {
  background-color: pink;
}
```

Testo che cambia lo sfondo se ci passo sopra e non ci sono mai passato prima

Link a cui cambia il background-color se è già stato visitato

Si possono specificare più pseudo classi contemporaneamente in un selettore come nell'esempio a:link:hover.

11.3.3 Selezione per contesto: first-child

Esistono anche pseudo classi che verificano il contesto dell'elemento corrente:

- first-child (il primo figlio del padre corrente, per esempio il primo <li> di una lista si indica on li:first-child)

Esempio di pseudo classe first-child

```
li:first-child {
  color : red;
  background-color : yellow
}
```

- Primo figlio - primo <li> all'interno del <ul> (o <ol>)
- Secondo figlio
- Terzo figlio

Altro esempio di pseudo classe first-child

```
<style type="text/css">
  p:first-child { text-indent: 2em; }
</style>

<body>

  <p>Questo è il testo del
primo paragrafo allungato
per andare a capo.</p>

  <p>Mentre questo è il testo
del secondo paragrafo con
indentazione iniziale.</p>

</body>
```

Questo è il testo del primo paragrafo allungato per andare a capo.
Mentro questo è il testo del secondo paragrafo con indentazione iniziale.

Nell'esempio solo il primo figlio del body di tipo `p` viene influenzato dalla regola.

11.3.4 First-letter, first-line

È anche possibile individuare la prima lettera di una parola (first-letter) ed la prima riga di un elemento (first-line).

Esempi di pseudo classi first-letter e first-line

```
#exampleFirstLetter:first-letter {
  text-transform : uppercase;
  font-size: 2em
}


#exampleFirstLine:first-line {
  color : red;
  font-style: italic
}
```

La prima lettera è impostata maiuscola e raddoppiata tramite lo style.

La prima riga di questo paragrafo comparirà con font-style "italic" e di colore rosso, mentre le altre righe avranno font-style e colore ereditato dal padre.

Le pseudo classi "first-line" e "first-letter" non funziono su tutti i browser. Verificate se sui browser che sono nel vostro target queste classi funzionano. La pseudo classe first-line permette tra le altre cose di essere verificata in “modo dinamico”, se io allargo la finestra del mio browser e la prima riga diventa più larga, la “nuova” prima riga sarà influenzata dalla dichiarazione soggetta al selettore :first-line.

11.3.5 Selettori: una tabella riassuntiva

Pattern	Matches
*	qualunque elemento
T	ogni elemento di tipo "T"
T1 T2	ogni elemento di tipo "T2" discendente di un elemento di tipo "T1"

.class	ogni elemento con class="class"
#id	l'elemento con id="id"
:first-child	ogni elemento che il primo figlio di suo padre
:link	ogni elemento che non è stato visitato
:visited	ogni elemento che è già stato visitato
:active	ogni elemento che è attivato dall'utente e l'azione non è ancora stata terminata
:hover	ogni elemento "ricoperto" dal mouse
:focus	ogni elemento su cui è il focus (che rivelerebbe il prossimo input da tastiera
x, y	raggruppare gli elementi che corrispondono a x **or** a y
T:first-letter	la prima lettera di ogni elemento di tipo "T"
T:first-line	la prima linea di ogni elemento di tipo "T"

11.4 Box Model

Il modello di formattazione visuale dei CSS è basato su contenitori rettangolari: i "box".

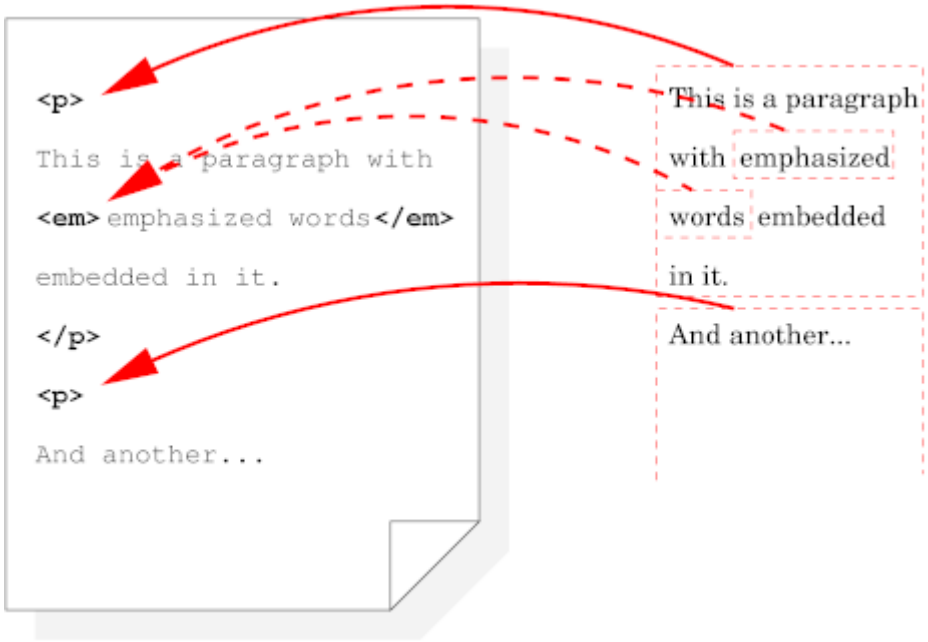


Figura 18 - Il Box Model

I box sono degli elementi indivisibili che forniscono una rappresentazione della struttura della pagina. Ad ogni elemento HTML corrisponde un box. L'innestamento dei blocchi corrisponde all'innestamento degli elementi HTML.

11.4.1 I tipi di box e la proprietà Display

Esistono diversi tipi di **box** che hanno comportamenti diversi, l'appartenenza ad un certo tipo può essere impostata tramite la proprietà `display`:

- display: block
- display: inline
- display: list-item
- display: table
- ...

- display: none

Impostare la proprietà display, purtroppo, non funziona su tutti i browser (in particolare non funziona con tutti gli elementi su explorer).

11.4.2 Display: block

Gli elementi di tipo block sono box rettangolari che cominciano e finiscono con un interruzione di linea.

Elementi che tipicamente sono di tipo block sono:

- div
- p
- blockquote
- h1

11.4.3 Display: inline

Gli elementi di tipo inline sono sempre box rettangolari che:

- **non** cominciano e **non** finiscono con un interruzione di linea.
- sono mostrati nella solita linea del contenuto precedente.
- hanno la dimensione che dipende dal proprio contenuto.

Elementi che tipicamente sono di tipo inline sono:

- span
- strong
- em
- acronym

11.4.4 Display:none

Impostando la proprietà `display` a `none`, l'elemento non verrà visualizzato.

`display: none` può essere utilizzato per interessanti effetti visivi, come per esempio menu a scomparsa in funzione di eventi (per esempio hover).

L'esempio seguente mostra un elemento – il div class=”descrizione” - che normalmente non viene mostrato (tramite il display: none), ma quando si va sopra (:hover) l'ancora, il div compare (display:block)

Esempio display:none

```
a.menu div.descrizione {
  display: none;
}
a.menu:hover div.descrizione {
  display: block;
  color: green;
}
```

```
<a name="organizzazione" class="menu">
   organizzazione
  <div class="descrizione">
    Sezione riguardante
l'organizzazione
    della società
  </div>
</a>
```

La prima regola CSS

```
a.menu div.descrizione {display: none}
```

fa si che l'elemento - `div.descrizione` - non sia visibile.

La seconda regola CSS

```
a.menu:hover div.descrizione {
  display: block; color: green;
}
```

fa si che l'elemento - `div.descrizione` - sia reso visibile solo quando ci passiamo sopra (`:hover`).

11.4.5 Display:list-item

- Gli elementi di tipo `list-item` (display:list-item) sono dei box con etichetta
- di solito sono utilizzati in HTML per rappresentare le liste.
- Gli elementi `li` sono un esempio di elementi `list-item`
- l'etichetta può essere o meno presente e può essere anche personalizzata.

11.5 Personalizzare le liste

11.5.1 Personalizzare le liste: list-style-type

È possibile personalizzare i "marcatori" delle liste, tramite la proprietà `list-style-type`.

I possibili valori sono riportati nella tabella sottostante insieme ad altre caratteristiche.

Nome:	list-style-type
Valori:	disc | circle | square | decimal | decimal-leading-zero | lower-roman | upper-roman | lower-greek | lower-alpha | lower-latin | upper-alpha | upper-latin | hebrew | armenian | georgian | cjk-ideographic | hiragana | katakana | hiragana-iroha | katakana-iroha | none | inherit
Valore Iniziale:	none
Si applica a:	elementi block-level
Acquisizione:	no
Percentuali:	N/A

Esempi di liste-style-type

```
<style type="text/css">
  ol {
    list-style-type: upper-roman;
  }

  ol ul {
    list-style-type: square;
  }
</style>

<ol>
  <li>uno</li>
  <li>due</li>
  <li>
    <ul>
      <li>cane</li>
      <li>gatto</li>
    </ul>
  </li>
</ol>
```

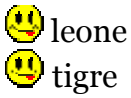
- uno
- due
  - cane
  - gatto

11.5.2 List-style-image

È possibile inserire un'immagine come etichetta delle liste tramite la proprietà `list-style-image`.

Esempi di list-style-image

```
<style type="text/css">
  ul {
    list-style-image: url("images/faccina.png");
  }
<style>

<ul>
  <li>leone</li>
  <li>tigre</li>
</ul>
```



11.5.3 List-style-position

Questa proprietà posiziona l'etichetta dentro o fuori il box che racchiude il testo.

Esempi di list-style-position

```
ul {
  list-style-position:
outside
}
```

- Primo elemento della lista, con etichetta posizionata fuori (outside)
- Secondo elemento della lista. Il testo si allinea sotto la prima lettera del box non sotto l'etichetta

```
ul {
  list-style-position: inside
}
```

- Primo elemento della lista, con etichetta posizionata dentro (inside)
- Secondo elemento della lista. Il testo si allinea sotto l'etichetta

11.5.4 List-style

La proprietà `list-style` permette di esprimere in maniera coincisa tutte le proprietà di tipo `list-style-...`

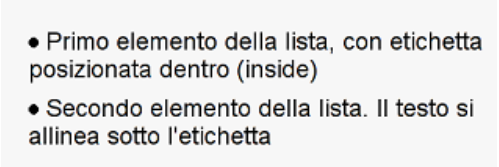
La sintassi è:

list-style: <list-style-type> | <list-style-image> | <list-style-position>

Esempio

- primo elemento
- secondo elemento

```
li.miaLista {
  list-style: disc inside
}
```



11.6 I colori: la proprietà color

La proprietà color controlla il colore del testo contenuto all'interno dell'elemento, questa proprietà è applicabile a tutti gli elementi ed è ereditata dai figli.

Il colore può essere espresso nelle tre forme:

- colori predefiniti -> green, blue, yellow, white, black,etc
- rgb -> rgb(0,255,0) - i tre numeri sono i componenti in Red Green Blue da 0 a 255 del colore
- esadecimale -> #00FFAA – la prima copia di numeri rappresenta la componente di Red (00), la seconda (FF) quella di Green, la terza(AA) quella di Blue, le componenti sono riportate in esadecimale da 00 a FF.

Esempi di colori

```
color: blue
```

Testo di colore blue

```
color: rgb(255, 0, 255)
```

Testo di colore viola

```
color: #00FF00
```

Testo di colore verde

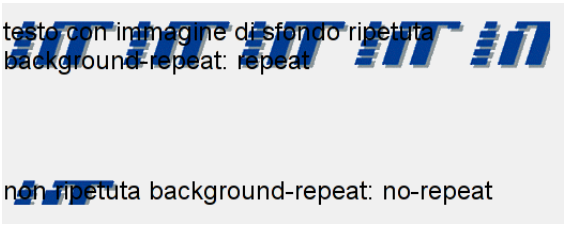
11.7 Background

11.7.1 background-image, background-repeat

In ogni elemento è possibile indicare una immagine di sfondo (`background-image`) e se e come ripetere questa immagine (`background-repeat`).

Esempi di background-image

```
<style type="text/css">
 div#backgroundRepeat {
  background-image:
url("images/faccina.png");
  background-repeat: repeat;
 }
 div#backgroundNoRepeat {
  background-image:
url("images/faccina.png");
  background-repeat: no-repeat;
}
<style>

<div id="backgroundRepeat">
 testo con immagine
 di sfondo ripetuta
</div>
<div id="backgroundNoRepeat">
 non ripetuta
</div>
```



11.7.2 background-position

`background-position` permette di specificare dove posizionare l'immagine di background.
Si possono utilizzare alcuni valori definiti:

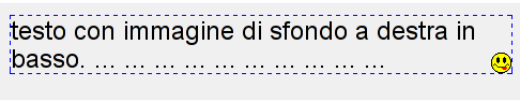
- top, left, right, center, bottom

Oppure combinazioni di questi.

Inoltre la posizione si può specificare con posizioni relative all'elemento corrente (esempio background-position: top corrisponde a 0% 0%)

Esempio di background-position

```
< style type="text/css">
  div {
    background-image:
url("images/faccina.png");
    background-position: bottom right;
  }
</style>

<div>
  testo con immagine
  di sfondo a destra in
  basso. ... ... ... ...
  ... ... ... ... ... ...
</div>
```



11.7.3 Proprietà background

La proprietà background imposta tutte le proprietà relative al background.

La sintassi è:

background:
<background-image> | <background-position> | <background-repeat> | <background-attachment> | <background-color>

L'ordine con cui sono specificate le varie *"sotto-proprietà"* non è importante.

Esempio

```
body {
  background: url(images/faccina.png) no-reapeat scroll bottom
right
}
```

background-attachment

Valore:	scroll | fixed | inherit
Initial:	scroll
Si applica a:	tutti gli elementi
Acquisizione:	no

La proprietà **`background-attachment`** specifica se l'immagine di sfondo deve essere fissa (**`fixed`**) rispetto al *viewport* (tipicamente la finestra del browser), oppure deve essere posizionata relativamente all'elemento che la contiene (**`scroll`**). Nel caso l'immagine sia fissa ed all'interno di un elemento che non sia **`html`** o **`body`** l'immagine dovrebbe essere visibile solo quando dentro la finestra di visibilità dell'elemento corrente.

Per maggiori informazioni sulla proprietà background-attachment si veda la specifica CSS2 http://www.w3.org/TR/CSS2/colors.html#propdef-background-attachment.

11.8 Box Model: dimensioni

Ogni box ha associato delle proprietà che ne determinano le dimensioni e le "distanze" da gli altri box.

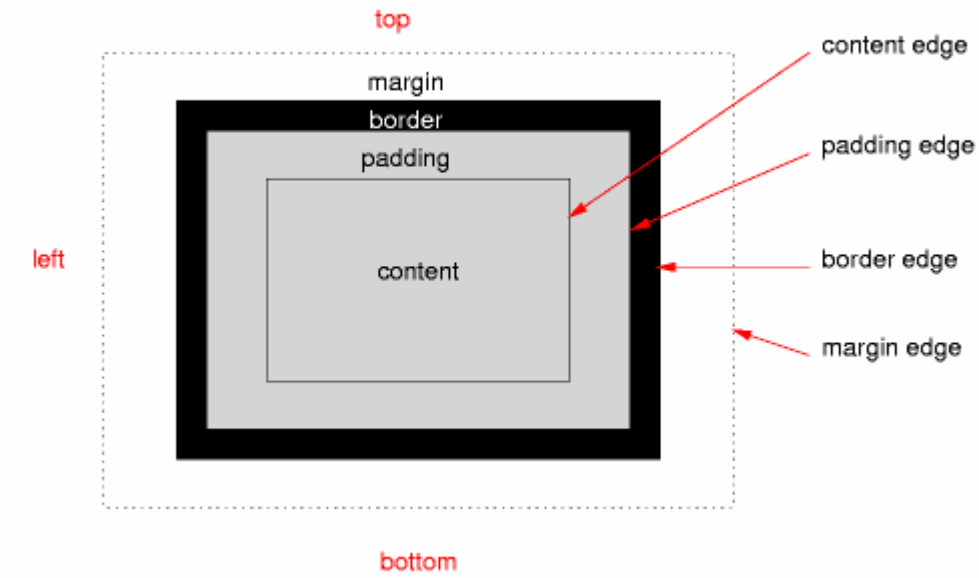


Figura 19 - Composizione del Box

Gli elementi che prendono parte alla formazione di un box sono:

- margin
- border
- padding
- width

Le dimensioni orizzontali di un box sono date dalla seguente formula:

larghezza box = margin-left + border-left-width + padding-left + width + margin-right + border-right-width + padding-right

primo DIV, con margini ma senza padding

secondo DIV, con margini e padding

- Il colore di sfondo tra il bordo ed il margine è trasparente.
- Il colore di sfondo tra il padding ed il bordo è quello dell'elemento corrente

11.9 Un utilizzo comune dei CSS: i margini, il padding ed i bordi

11.9.1 I margini

Un'altra delle proprietà più frequenti da impostare tramite i CSS sono i margini.

Attraverso le proprietà margin-[top, right, bottom, left] è possibile specificare lo spazio da lasciare intorno agli elementi.

I margini impostabili sono:

- margin-top
- margin-right
- margin-bottom
- margin-left

Le unità di misura utilizzabili sono quelle già descritte per i font, nel caso di unità espresse in percentuale, la percentuale si riferisce alla dimensione dell'elemento che contiene quello corrente.

Esempio di utilizzo dei margini

```
<style type="text/css">
  blockquote {
    margin-top: 1em;
    margin-right: 0em;
    margin-bottom: 1em;
    margin-left: 2em;
    background: pink;
  }
<style>


Testo normale

<blockquote>
 Testo indentato dentro il blockquote
</blockquote>

Testo normale
```

Testo normale

Testo indentato dentro il blockquote

Testo normale

11.9.2 Proprietà margin

La proprietà `margin` permette di specificare in modo coinciso tutti e quattro i margini di un elemento. La proprietà `margin` può essere espressa nel forme seguenti:

margin: 1em
: Tutti e quattro margini sono impostati ad 1em

margin: 1em 2em
: I margin top e bottom sono impostati ad 1em, i margin left e right sono 2em

margin: 1em 2em 3em
: margin-top: 1em, margin-left, margin-right:2em, margin-bottom:3em

margin: 1em 2em 3em 4em
: I margini sono impostati in senso orario partendo dal margin-top: margin-top: 1em, margin-left: 2em, margin-bottom: 3em, margin-right: 4em

11.9.3 padding

Si possono impostare le distanze tra il contenuto di un elemento ed il suo bordo tramite le proprietà di padding.

Le proprietà impostabili sono:

- padding-top
- padding-right
- padding-bottom
- padding-left

È possibile, come per i margini, specificare tutti i padding con una sola proprietà: `padding`.

Le regole per la proprietà `padding` sono le solite della proprietà `margin`.

Esempio di padding

```
<style type="text/css">
  p {
    padding-top: 1em;
    padding-right: 1em;
    padding-bottom: 1em;
    padding-left: 3em;
    background: pink;
  }
<style>

Testo normale

<p>
 Testo con padding dentro
 il paragrafo
</p>

Testo normale
```

Testo normale

Testo con padding dentro il paragrafo

Testo normale

11.9.4 I Bordi

La presentazione dei bordi di un box può essere controllata tramite diversi parametri:

- la larghezza (border-width)
- il colore (border-color)
- lo style (border-style)

Ognuno di questi parametri può essere controllato sui quattro lati del box:

- top, right, bottom, left

11.9.5 border-width

La proprietà `border-width` controlla la larghezza del bordo del box e raggruppa (allo stesso modo di `margin` e `border`) le quattro proprietà:

- border-top-width
- border-right-width
- border-bottom-width
- border-left-width

Il valore della proprietà è specificato con le solite unità di misura: em, ex, px, etc.

Esempi di border-width

```
p { border-width: 0.2em }

p { border-right-width: 0.1em }
```

Div con bordo di 0.2em

div con bordo destro di 0.1em

11.9.6 border-color

La proprietà `border-color` controlla il colore del bordo. `border-color` raggruppa (allo stesso modo di `margin` e `border`, etc.) le quattro proprietà:

border-top-color, border-right-color, border-bottom-color, border-left-color

Il valore della proprietà è espresso con i soliti metodi della proprietà color: **rgb**, **valori predefiniti**, **esadecimale**.

Esempi di border-color

border-color: blue	Div con bordo blue
border-color: rgb(255,0,0)	Div con bordo rosso -> rgb(255,0,0)
border-color: #000	Div con bordo nero -> #FFF

Se queste slide sono stampate in bianco e nero, negli esempi sopra riportati non si vedrà la differenza dei colori: usate un pò di fantasia.

11.9.7 border-style

La proprietà `border-style` controlla lo stile del bordo. `border-style` raggruppa (allo stesso modo di `border-color`) le quattro proprietà:

border-top-style, border-right-style, border-bottom-style, border-left-style

Figura 20 - Possibili valori della proprietà border-style

11.9.8 Proprietà border

La proprietà `border` racchiude tutte le proprietà che abbiamo fino ad ora visto sui bordi.

Se si vogliono esprimere valori diversi per i vari bordi valgono le regole viste per le altre proprietà riassuntive come `margin` e `border`.

I valori passati vengono associati alle varie proprietà disponibili per i bordi, per esempio impostando border: 1pt viene associato al border-width il valore di 1pt.

Esempi di utilizzo della proprietà border

`border: 1pt 2pt`	imposta border-top-width:1, border-right-width:1 .....
`border: 1pt solid green 2pt dashed red`	imposta i bordi top e bottom a 1pt solid green, i bordi left e right a 2pt dashed red

Allo stesso modo della proprietà `border` esistono anche le proprietà:

- border-top
- border-right
- border-bottom
- border-left

11.10 width, height

Abbiamo visto che le dimensioni orizzontali di un box sono date dalla formula:

larghezza box = margin-left + border-left-width + padding-left + width + margin-right + border-right-width + padding-right

Le dimensioni verticali sono date dall'analoga formula:

altezza box = margin-top + border-top-width + padding-top + height + margin-bottom + border-bottom-width + padding-bottom

- le proprietà `width` e `height` controllano le dimensioni del contenuto di un box.
- queste proprietà sono utilizzate con gli elementi **block-level** e gli elementi **replaced** (tipo le immagini).
- sugli elementi di tipo inline non hanno valenza.
- i valori delle proprietà sono esprimibili con le solite unità di misura oppure tramite il valore `auto`

Esempio di width ed height

```
div.mioBox {
  width: 9em;
  height: 9em;
  border: 1pt solid blue;
}
```

Div di dimensioni 9em x 9em. Lo spazio non occupato dal testo viene trattato come padding

11.10.1 width, margin-left, margin-right: auto

Il valore auto è il valore predefinito della proprietà `width` ed anche delle proprietà `margin-left` e `margin-right`.

Il suo significato cambia in funzione del tipo di elemento a cui è applicato, la tabella sottostante ne riassume il significato.

	elementi inline		elementi block	
	replaced	**non-replaced**	**replaced**	**non-replaced**
width	larghezza intrinsica	N/A	larghezza intrinsica	massimizza
margin-left, margin-right	0	0	massimizza	massimizza

Gli elementi di tipo **replaced** sono quelli il cui valore sostituisce l'elemento corrente, come per esempio le immagini.

Con **larghezza intrinsica** si intende la dimensione dell'oggetto, per esempio la larghezza intrinsica di un elemento `img` è la larghezza dell'immagine stessa.

Con **massimizza** si intende la massima dimensione disponibile all'interno dell'elemento padre.

Ci sono 3 possibili scenari di proprietà width, margin-left, margin-right ad auto che sfruttano il meccanismo della "massimizzazione":

uno dei valori è auto
: in questo caso il valore viene impostato al massimo valore possibile

nessuno dei tre valori è auto
: in questo caso viene ignorato il valore della proprietà margin-right e viene ricalcolato in funzione dello spazio disponibile.

due o tre valori sono auto ed uno di questi è width
: prima di tutto viene massimizzata la proprietà width fino al massimo valore disponibile, i margini ad auto diventano zero.

width è fissata ed i margini sono entrambi ad auto
: i margini vengono imposti uguali ed alla massima dimensione disponibile

E se la width è impostata ad una larghezza maggiore rispetto all'elemento padre?
Il comportamento è dettato dalla proprietà `overflow`, per maggiori informazioni http://www.w3.org/TR/CSS2/visufx.html#x0**.**

11.11 Schemi di posizionamento nei CSS

È possibile disporre i box secondo tre schemi di posizionamento:

- *Normal flow* dove il flusso normale prevede la disposizione sequenziale dei blocchi di testo, inline box, liste, eccetera.
- *Float* dove la posizione dei blocchi può essere spostata, il più possibile, a destra o a sinistra nel contesto corrente.
- *Absolute positioning* dove la posizione di un box è rimossa dal normale flusso e posizionata in funzione di un box che lo contiene.

11.11.1 Posizionamento degli elementi

Si possono posizionare i box indipendentemente dal normale flusso (dedotto dalla struttura) dell'HTML.

Nome:	position
Valori:	<static> | <relative> | <absolute> | <fixed> | <inherit>
Valore Iniziale:	static
Si applica a:	tutti gli elementi
Acquisizione:	no

Il posizionamento *static* è quello predefinito ed è quello che abbiamo usato fin qui.

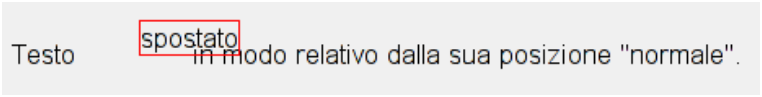
11.11.2 Posizionamento relativo

Impostando la proprietà CSS **`position`** `al valore` **`relative`** si indica che il box sottoposto a quella regola sia spostamento relativamente dalla posizione che il box avrebbe assunto normalmente. Le posizione degli altri box non sono influenzate da quella indicata per questo box, anzi gli altri box si comportano come se il box fosse nella sua posizione originale. Un effetto collaterale è che indicando uno spostamento si può, per esempio, far sovrapporre questo box ad altri.

Gli spostamenti si indicano con uno o più tra le proprietà **`top`**, **`bottom`**, **`left`** e **`right`**.

Esempio di posizionamento relativo

```
span {
  position: relative;
  top: -10px;
  left: 2em;
  border: 1px solid red;
}

Testo <span>spostato</span>
in modo relativo dalla sua
posizione "normale".
```



Come già accennato, nel calcolo degli spazi occupati dal box posizionato in modo relativo si usa il posizionamento normale (`static`). Il posizionamento reale però sarà quello indicato dalle proprietà `top, bottom, left, right.`

11.11.3 Posizionamento fisso

Nel posizionamento fisso il box è separato dal normale flusso del documento e viene posizionato alla coordinate indicate con le proprietà `top, bottom, left, right.` Il box sottoposto a questa regola è fisso nel *viewport (nello schermo) e può permettere di avere dei risultati visivi simili a quelli dei* `frame`.

Esempio di layout fisso.

```
position: fixed;
width: 100%;
height: 25%;
left: 0;
top: 0;
right: 0;
bottom: auto;
border: 1px solid gray;
```

```
position: fixed;
width: 25%;
height: 75%;
left: 0;
top: 25%;
right: auto;
bottom: 0;
background-color: ligh
border: 1px solid gray
```

```
position: fixed;
width: auto;
height: 75%;
left: 25%;
top: 25%;
right: 0;
bottom: 0;
background-color: lightyellow;
border: 1px solid gray;
```

Non tutti i browser supportano il posizionamento fisso, ad esempio Internet Explorer non la supporta fino alla versione 6.

11.11.4 Posizionamento assoluto

Usando il posizionamento fisso il box esce dal normale flusso e la posizione viene indicata rispetto al primo elemento padre (o comunque precedente nell'albero HTML) posizionato. Se non ci sono elementi posizionati che contengono l'elemento a cui questa regola viene applicata, la posizione viene calcolata rispetto al *viewport (lo schermo)*.

Con "primo elemento padre posizionato" si intende il primo elemento che ha impostata la proprietà **position** ad un valore diverso da **static** e che contiene l'elemento attuale.

Ecco un esempio di codice per il posizionamento assoluto:

```
<div style="position: relative">
  1. questo è posizionato

  <div>
    2. non posizionato (direttamente, ma il padre sì)

    <div style="position: absolute;">
      3. posizionato assolutamente rispetto ad 1.
    </div>
  </div>
</div>
```

Esempio di posizionamento relativo:

```
<p style="position: relative; margin-right: 7em; left: 5em;">

 Questo paragrafo ha il nome dell'autore disposto a sinistra e
alla fine del paragrafo stesso.
 <span
  style="position: absolute; bottom: 0; left: -5em; color: red;">
  Marco
 </span>
<p>
```

Marco Questo paragrafo ha il nome dell'autore (in rosso) messo sulla sinistra del paragrafo stesso.

11.11.4.1 Float

La proprietà **float** indica se il box deve essere allineato il più possibile a sinistra o destra. Ecco un riassunto delle sue caratteristiche:

Valore:	left | right | none | inherit
Valore iniziale:	none
Si applica a:	tutti gli elementi esclusi quelli posizionati e quelli generati
Acquisizione:	no

Impostando la proprietà **float** ad un valore diverso dal valore predefinito **none** si chiede al browser di interpretare l'elemento secondo lo schema di posizionamento *float* (cioè flottante/volante). È da notare che questo schema è separato da quello di posizionamento normale (normal flow) e che quindi va posta particolare attenzione per il suo uso.

Esempi di float (left e right):

```
<img style="float: left"
 src="images/floating.png"
 alt="Esempio di immagine float"
/>
Non hai veramente ...
```

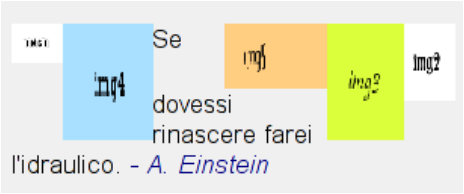
Non hai veramente capito qualcosa fino a quando non sei in grado di spiegarlo a tua nonna. A. Einstein

```
<img style="float: right"
 src="images/floating.png"
 alt="Esempio di immagine float"
/>
I computer sanno ...
```

I computer sanno contare solo da 0 a 1, il resto è illusione.

Esempio di float con dimensioni e margini:

```
<style type="text/css">
 p { width: 24em }
 #i1 { float: left; width: 2em;
height: 1.5em }
 #i2 { float: left; width: 2em;
height: 3em }
 #i3 { float: right; width: 3em;
height: 4.5em }
 #i4 { float: left; width: 3.5em;
height: 4.5em }
 #i5 { float: right; width: 2.5em;
height: 1.5em }
</style>
<img id="i1" src="i1.png" alt="i1"
/>
<img id="i2" src="i2.png" alt="i2"
/>
<img id="i3" src="i3.png" alt="i3"
/>
<img id="i4" src="i4.png" alt="i4"
/>
<img id="i5" src="i5.png" alt="i5"
/>
Se dovessi rinascere farei
l'idraulico.
```



11.11.4.2 Clear

La proprietà `clear` indica quale lato del box di un elemento non deve essere adiacente con altri box flottanti. Ecco un riassunto delle sue caratteristiche:

Nome:	clear
Valori:	<none> | <left> | <right> | <both> | <inherit>
Valore Iniziale:	none
Si applica a:	elementi block-level
Acquisizione:	no

Esempi d'uso di clear:

```
<p>primo paragrafo</p>
<img style="float: left;" ...
<p>secondo
paragrafo</p>
```

Primo paragrafo

Secondo paragrafo

```
<p>primo paragrafo</p>
<img style="float: left" ...
<p style="clear:left">secondo
paragrafo</p>
```

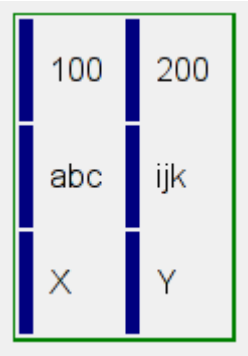
primo paragrafo

secondo paragrafo

11.12 Tabelle

Le proprietà gia viste per gli altri box si applicano in larga misura anche agli elementi delle tabelle (margini, bordi, sfondi...). Ad esempio ecco una tabella con l'uso di CSS:

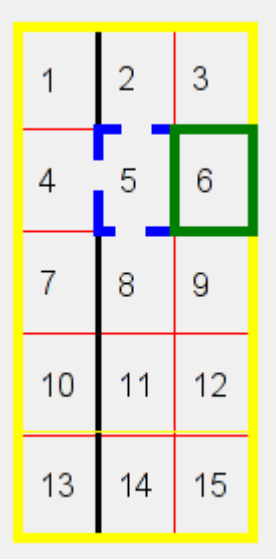
```
table {
  border: 1pt solid green;
}

td {
  border-left: 5pt solid darkblue;
}
```



Alcune proprietà delle tabelle sono supplementari e permettono di definire la distanza tra le varie celle. Normalmente i bordi delle varie celle sono separati: è possibile impostare separatamente ciascuno dei bordi su ogni cella singolarmente. È però possibile impostare un modello collassato dei bordi in cui tutti i bordi sono uniti (sia di celle che di colonne e righe). La proprietà in questione è la `border-collapse` che è usabile solo sulle tabelle (in teoria sul display delle tabelle).

Ecco un esempio sui bordi delle tabelle:

```
table {
  border-collapse: collapse;
  border: 5px solid yellow
}
*#col1 {
  border: 3px solid black
}
td {
  border: 1px solid red;
  padding: 0.5em
}
td.solid-blue {
  border: 5px dashed blue
}
td.solid-green {
  border: 5px solid green
}
```



L'argomento tabelle con i CSS non si esaurisce qui. In effetti il *table model* indica molte alte proprietà e modifica il comportamento di visualizzazione in modo sensibile rispetto al *block model*. Il riferimento per questo argomento è il W3C con Tabelle con CSS2 (http://www.w3.org/TR/CSS21/tables.html)

11.13 Gli altri media

I CSS (in particolare dalla versione 2 in poi) hanno la possibilità di indicare la presentazione anche su altri media rispetto al comune video. Esempi ne sono:

- aural/speech
- braille
- print
- screen
- tv.

Non trattaremo in modo specifico le proprietà di ciascun media, ma le indicazioni generali e le regole fornite fino a qui sono comunque valide anche per questi media.

11.13.1.1 Riferimenti per i CSS

- Web Style Sheets Home page http://www.w3.org/Style/
- Cascading Style Sheets, level 2 revision 1 (CSS 2.1) http://www.w3.org/TR/CSS21/
- CSS 3 «under construction» http://www.w3.org/Style/CSS/current-work

Cosa si fa e cosa non si dovrebbe fare

12.1 Ancora sulle tabelle

Per migliorare l'accessibilità della pagina è importante usare almeno gli elementi `th` e `caption` all'interno delle tabelle. In particolare il `th` dovrebbe essere usato per racchiudere le celle delle intestazioni, così da separarle dalle celle dei dati. L'elemento `caption` invece dovrebbe racchiudere un titolo quanto più descrittivo della tabella. I vari tipi di browser possono sfruttare questi due elementi per migliorare la presentazione (e molto spesso anche l'accessibilità) delle tabelle.

Ecco un esempio di uso di `th` e `caption` con la presentazione associata del browser:

```
<table>
 <caption>numero dei giorni>
 <tr>
  <th>mesi</th>
  <th>giorni</th>
 </tr>
 <tr>
  <td>gennaio</td>
  <td>31</td>
 </tr>
 <tr>
  <td>febbraio</td>
  <td>28 (o 29)</td>
 </tr>
</table>
```

numero dei giorni

mesi	giorni
gennaio	31
febbraio	28 (o 29)

Nota: l'uso di dimensioni fisse nelle tabelle, ma più in generale in tutti elementi, impedisce il ridimensionamento dei caratteri e dei box, ciò nel migliore dei casi impedisce la presentazione dei contenuti in modo proporzionale alle dimensioni reali della finestra e nel peggiore dei casi rende il contenuto poco leggibile. L*a dimensione media degli schermi cambia molto in fretta e una buona progettazione forse dovrebbe tenerne conto se la longevità del progetto è un punto importante.*

12.1.1 Quando non usare le tabelle

Le tabelle vengono ad oggi anche usate per dividere in parti la pagina per fare in modo che il testo sia posizionato in una certa posizione. Come già illustrato in precedenza questo può essere fatto con l'uso di CSS, separando così la presentazione dai contenuti.

Adesso vediamo alcuni esempi con cui usando i CSS si realizzano componenti grafiche riusabili.

12.2 Costruire il layout con stylesheet

I benefici dell'uso di stylesheet sono già stati elencati anche in precedenza, è bene comunque tenere presente che:

- si possono posizionare i box ovunque nella pagina;
- si possono allineare i box come si desidera;
- i box contenenti testo, immagini, ... possono presentare meglio di una tabella i contenuti di una pagina, non avendo legami con la struttura;
- alcune particolari interpretazioni delle specifiche dei CSS fa si che si debbano cercare delle soluzioni e provarle su più browser per avere maggior certezza (e correttezza).

Vediamo alcuni esempi.

12.2.1 Menu in CSS

Uno tra i metodi più classici per navigare il web è il classico menu. Di solito viene ingabbiato con una tabella vediamone una versione che usa i CSS (e non le tabelle).

Codice HTML:

```
<div class="menu">
  <a href="#">prima voce <span>dettagli 1</span></a>
  <a href="#">seconda voce <span>dettagli 2</span></a>
  <a href="#">terza voce <span>dettagli 3</span></a>
  <a href="#">quarta voce <span>dettagli 4</span></a>
  <a href="#">quinta voce <span>dettagli 5</span></a>
  <a href="#">sesta voce <span>dettagli 6</span></a>
  <a href="#">settima voce <span>dettagli 7</span></a>
  <a href="#">ottava voce <span>dettagli 8</span></a>
  <a href="#">nona voce <span>dettagli 9</span></a>
</div>
```

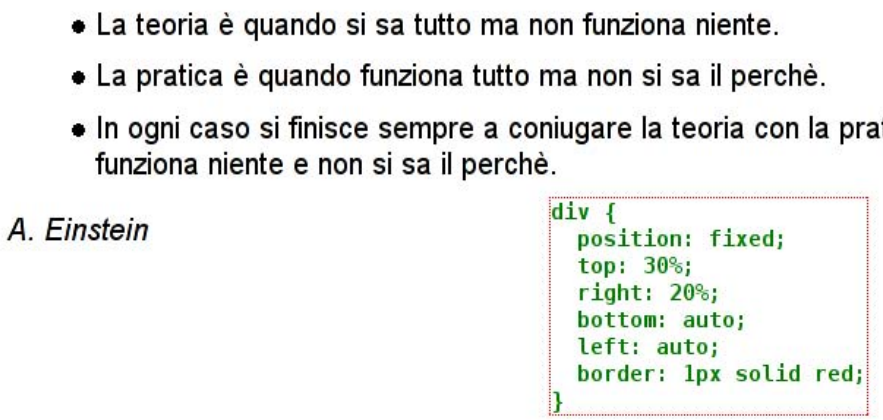
Codice CSS e immagine del risultato in un browser:

```
div.menu {
  float: left;
  width: 25%;
  background-color: lightcyan;
}

div.menu a {
  position: relative;
  display: block;
  margin: 0;
  padding: 1em;
  border: 1px solid darkblue;
}

div.menu a:hover {
  background-color: lightred;
  color: darkred;
}

div.menu a span {
  display: none;
}

div.menu a:hover span {
  position: absolute;
  top: 0;
  left: 100%;
  width: 200px;
  border: 1px solid red;
  background-color: lightgreen;
  display: block;
  z-index: 100
}

div.contenuto {
  float: left;
  width: 70%;
}
```

Nell'immagine si nota che la seconda voce è evidenziata – a causa del mouse hover – e di conseguenza è visibile anche il contenuto dell'elemento **span**, i dettagli.

12.2.2 Posizionamento e layout con i CSS

Nell'immagine ecco un elemento `div` posizionato in mezzo alla pagina. Le posizioni sono indicate in modo proporzionale alla pagina e si usa il posizionamento `fixed`).



Ecco un esempio di un tipico layout Web realizzato con CSS:

In caso il browser non supporti gli stylesheet - come Lynx, vecchi browser grafici, o browser di altra natura - oppure non ne supporti alcune caratteristiche sarà comunque riportato il testo nell'ordine con cui appare nel codice HTML. Quindi è meglio far sì che l'ordine con cui appare il testo nella pagina HTML sia anche l'ordine logico del documento.

Anche la verifica delle pagine con più browser come già detto, può dare buoni risultati.

12.2.2.1 Approfondimenti sul posizionamento e i CSS

- http://www.csszengarden.com/ dimostra che l'uso dei CSS non sia riduttivo nel rendere pagine grafiche accattivanti e usando standard WAI (accessibilità) e XHTML.
- http://www.positioniseverything.net/ contiene una serie di articoli - aggiornati - sul posizionamento via CSS e i problemi con i vari browser.
- http://www.literarymoose.info/ un bell'esempio di uso di CSS.
- http://www.designdetector.com/tips.php suggerimenti ed altri esempi di CSS.

Progettare un sito: parte III

13.1 Cosa è l'accessibilità

Nella sua accezione terminologica principale l'accessibilità è identificata con la possibilità data ad utenti disabili di poter accedere ad un qualsiasi prodotto. Nel campo della comunicazione via Web garantire l'accessibilità significa progettare le pagine tenendo conto di alcune caratteristiche che permettano anche ad utenti con disabilità di poter accedere al sito[43]. Ai diversi gradi di disabilità possono corrispondere diversi livelli di progettazione, come vedremo di seguito.

Tenere in considerazione gli aspetti di accessibilità di un sito web è molto importante perchè:

- riguarda *tutti, e* non solo persone con problemi fisici, mentali o psichici, ma ... anziani, bambini, persone con poca conoscenza informatica, noi stessi, in condizioni particolari date da disabilità temporanee o incidenti più o meno gravi
- e *tutto*, browser diversi, piattaforme diverse, dispositivi diversi (TV, monitor, PDA, telefonino, ecc..), risoluzioni diverse, informazioni non comprensibili
- è una parte dell'*usabilità (cfr. § 1.1.3)*

Come riportato nella parte introduttiva della Raccomandazione del W3C "Web Content Accessibility Guidelines 1.0" del 5 maggio 1999, le difficoltà nell'accessibilità riguardano in genere tutte quelle categorie di persone che potrebbero:

- avere disturbi alla vista, all'udito, al movimento;
- avere difficoltà nella lettura o comprensione del testo;
- avere difficoltà nell'utilizzo delle periferiche standard di input: mouse o tastiera;
- avere solo un browser di tipo testuale, o uno schermo di piccole dimensioni o una connessione lenta ad Internet;
- non comprendere completamente la lingua con cui è stato scritto il testo;
- essere in situazioni tali da avere impegnati gli occhi, le orecchie o le mani (es.: guidare, lavorare in un ambiente rumoroso, etc.);
- avere una versione non aggiornata del browser, o un browser completamente diverso, un browser vocale, o un differente sistema operativo.

[43] Per un approfondimento vedere la serie di linee guida prodotte dal gruppo di lavoro Web Accessibility Initiative del World Wide Web Consortium-W3C. (http://www.w3.org/WAI)

Accessibilità non si riferisce quindi esclusivamente al “facile accesso”, ma anche ad una “facile comprensione”. Accessibile non deve essere soltanto la tecnologia, ma anche l’informazione che viene trasmessa. A questo proposito, dal vocabolario della lingua italiana lo Zingarelli, ed. 1997, leggiamo le seguenti definizioni del termine “accessibile”:

- di facile accesso: *luogo, strada a.; località non a. d'inverno.* SIN. Raggiungibile
- (fig.) di facile comprensione: *concetto, idea a.; cognizioni accessibili a tutti*

In questa seconda accezione, l’accessibilità come fruibilità dell’informazione è legata al modo in cui vengono scritti i testi per il web[44] o al modo stesso in cui vengono inseriti elementi visuali comprensivi di una propria simbologia.

Oltre alla lingua usata sul web, molti studi sono stati svolti proprio sul linguaggio testuale e su come esso viene decifrato e interpretato dagli utenti. Ciò che è emerso con forza è che le persone "non leggono, ma scorrono il testo":."[...] they don't actually read: instead, they scan the text[45]". Inoltre, come è noto dagli studi di *Human Computer Interaction*, la lettura a schermo è ridotta del 25% rispetto alla lettura su foglio stampato; entrano in gioco variabili diverse, quali il tipo di schermo e la sua risoluzione, il tipo di carattere e la sua grandezza, il contrasto fra il colore del testo e il colore dello sfondo, l'ambiente circostante (rumore, illuminazione, etc.)[46] . Tutti questi fattori, insieme alla "leggibilità" stessa del testo, possono contribuire alla perdita di concentrazione e quindi all’abbandono della pagina in caso di eccessive difficoltà.”

13.1.1 L’ISO e l’accessibilità

L’International Organization for Standardisation (ISO) anche nel caso dell’accessibilità, ha prodotto una specifica tecnica per la progettazione di software che sia più facile da usare da anziani, disabili, ma in generale per tutte le persone, molto simile alle regole già distribuite dal W3C.

Il documento, ISO/TS 16071, fornisce una guida per la progettazione di software che debba tenere in considerazione le diverse capacità degli utenti, in modo da assicurare che possano essere utilizzati anche da persone con problemi visivi, uditivi, motori o cognitivi, ma anche dagli anziani, dai disabili temporali e da tutti coloro che non hanno alcun tipo di disabilità.

---

[44] Per approfondimenti sulla scrittura per il web, cfr. http://www.sun.com/980713/webwriting/, la guida ufficiale della Sun Microsystems su "Writing for the Web". Cfr. inoltre gli articoli di Nielsen all’indirizzo http://www.useit.com/papers/webwriting/.

[45] John Morkes and Jakob Nielsen, *"Concise, SCANNABLE, and Objective: How to Write for the Web"*, http://www.useit.com/papers/webwriting/writing.html, 1997

[46] Cfr. Hulme Charles, "*Extracting Information From Printed and Electronically Presented Text*" in Andrew Monk "Fundamentals of Human Computer Interaction", Academic Press, Inc., 1984

"This standard is part of a series of standards aimed at making systems more usable, in particular, more effective, more efficient and more satisfying for all users - with either permanent or temporary disabilities," said Tom Stewart, Chair of the committee that developed the new standard. "We are not solely concerned with ensuring that users with disabilities are treated the same as other users. We want all users to be able to live and work in harmony with their computer systems."

La nuova specifica tecnica ISO/TS 16071 è uscita il 12 maggio 2003 con il nome "Ergonomics of human-system interaction - Guidance on accessibility for human-computer interfaces", ancora in forma di Technical Specification che potrebbe essere rivisto per poi essere pubblicato come Standard ufficiale.

13.1.2 Il W3C e l'accessibilità del web

"The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect." [Tim Berners-Lee, W3C Director and inventor of the World Wide Web]

Il World Wide Web Consortium si è occupato di problematiche legate all'accessibilità del web, sin dal 7 aprile 1997, giorno in cui a Cambridge Massachusetts, USA, ha annunciato la nascita del Web Accessibility Initiative (WAI)[47]. Lo scopo del WAI è quello di promuovere e raggiungere una funzionalità del web adatta a persone con disabilità. Sottoscritto inizialmente dalla Casa Bianca e dai Membri del W3C, il WAI ha avuto la responsabilità dello sviluppo dei protocolli software e delle tecnologie, creando le linee guida per l'uso delle tecnologie, sensibilizzando le industrie e svolgendo attività di sviluppo e ricerca.

Rendere il web accessibile significa permettere a persone diversamente abili di percepire, comprendere, navigare, interagire e partecipare attivamente al web. Come già accennato risolvere i problemi di accessibilità sicuramente beneficia anche altre categorie di utenza. Pensiamo oggigiorno alle persone anziane, sempre più numerose nelle società moderne, le cui abilità sicuramente cambiano con l'avanzare dell'età.

Il W3C ha creato un memorandum rapido in 10 punti che riassume i concetti chiave sull'accessibilità. I 10 punti vengono detti "Quick Tips" e non devono comunque essere considerati come linee-guida complete. I "Quick Tips" sono disponibili in più lingue, come una cartolina di riferimento delle dimensioni di un biglietto da visita, e possono essere richieste gratuitamente allo stesso Consortium.

Di seguito riportiamo il contenuto dei 10 punti delle "Quick Tips", in lingua italiana[48]:

[47] Per saperne di più sul lavoro del gruppo vedere all'indirizzo http://www.w3.org/WAI/

[48] Reperibili all'indirizzo http://www.w3.org/WAI/References/QuickTips/qt.it.htm

- **Immagini ed animazioni**. Utilizzare l'attributo **alt** per descrivere la funzione di ogni elemento grafico.
- **Immagini cliccabili**. Utilizzare l'elemento **map** e descrivere le zone attive.
- **Multimedia**. Fornire sottotitoli e trascrizioni per l'audio, e descrizione di filmati.
- **Link ipertestuali**. Utilizzare enunciati che conservino il loro senso al di fuori del contesto. Per esempio, evitare «cliccare qui».
- **Organizzazione**. Utilizzare titoli, liste e una struttura coerente. Utilizzare **CSS** per l'impaginazione.
- **Figure e diagrammi**. Descriverli all'interno della pagina o utilizzare l'attributo **longdesc**.
- **Script, applet e plug-in**. Fornire una pagina alternativa quando tali funzionalitá sono inaccessibili o non supportati.
- **Cornici (frames)**. Utilizzare **noframes** e titoli sigificativi.
- **Tabelle**. Facilitare la lettura linea per linea. Riassumere.
- **Verificare il lavoro**. Validare. Utilizzare gli strumenti, la lista di controllo e le linee guida di: http://www.w3.org/TR/WCAG.

13.1.3 Usabilità e accessibilità

Usabilità e accessibilità sono due concetti importanti per capire soprattutto l'evoluzione delle nuove tecnologie nei campi dell'informazione e della comunicazione. La cosa fondamentale è non confondere i due termini, come spesso accade.

Vi è stata infatti sempre molta confusione fra i due termini, ancora oggi spesso difficile da eliminare. Dobbiamo innanzitutto ricordare che, mentre l'usabilità si preoccupa di tutti gli utenti potenziali di un certo prodotto e li pone al centro del processo di progettazione, sin dalle prime fasi, l'accessibilità si preoccupa di un sotto insieme di utenti potenziali, quelli che potrebbero avere problemi derivanti da diversi tipi di disabilità.

Se quindi un sito viene progettato rispettando i criteri per l'accessibilità, questo non può assolutamente essere un requisito per affermare che il sito in questione è anche usabile. L'usabilità prevede verifiche, test e criteri di progettazione molto diversi da quelli previsti dall'accessibilità, che devono essere appositamente programmati dagli sviluppatori.

Non è quindi assolutamente scontato che un sito accessibile sia usabile, se non si sono previsti interventi specifici atti a coinvolgere direttamente gli utenti interessati.

13.2 Cosa sono le WCAG

WCAG sta per Web Content Accessibility Guidelines e sono documenti prodotti dal WAI-W3C in cui si spiega come è possible rendere il contenuto del web accessibile a persone diversamente abili. Per contenuto viene inteso qualsiasi tipo di informazione, da quella testuale, alle immagini, i suoni e altro ancora.

Le WCAG comprendono anche le Authoring Tool Accessibility Guidelines (ATAG) e le User Agent Accessibility Guidelines (UAAG). Vi sono diverse componenti che entrano in gioco quando si vuole creare contenuto accessibile per il web:

- il contenuto: ovvero l'informazione in una pagina web o in una applicazione web che comprende sia testo, immagini e suoni, sia il codice che definisce la struttura e la presentazione dell'informazione stessa
- i diversi browser o media player
- le tecnologie assistive come gli screen-reader, le tastiere alternative, ecc.
- il grado di esperienza che l'utente ha delle tecnologie che usa
- gli sviluppatori, siano essi designers, autori, sviluppatori di codice
- i programmi per lo sviluppo del codice
- i programmi per la valutazione del codice, come i validatori per l'HTML, i validatori per il CSS, ecc..

Tutte queste componenti stanno in relazione fra di loro e determinano l'accessibilità dell'informazione su web.

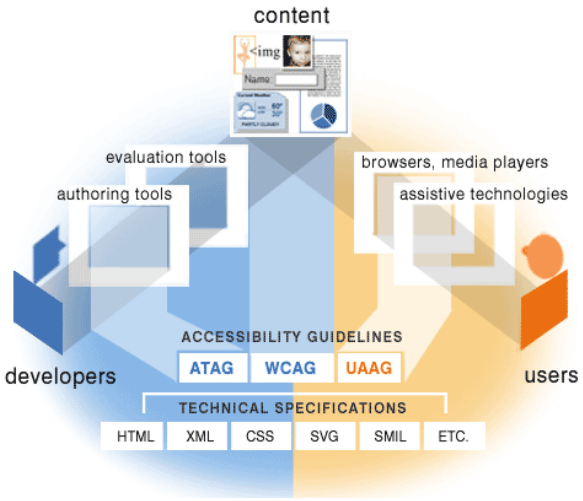


Figure 22 – Linee-guida per le diverse componenti

La figura 1 mostra le linee-guida per le diverse componenti:

- ATAG (Authoring Tool Accessibility Guidelines): sono specifiche per i programmi di sviluppo
- WCAG (Web Content Accessibility Guidelines): vengono usate dagli sviluppatori, dai programmi di sviluppo e dai programmi di validazione
- UAAG (User Agent Accessibility Guidelines): si riferiscono principalmente ai browser e ai media player, ma hanno anche qualche riferimento per le tecnologie assistive.

Tutte le linee-guida del WAI si basano sulle specifiche tecniche sviluppate per il web riguardanti per esempio i linguaggi HTML, XML, CSS, SVG, ecc..

13.2.1 La WCAG Recommendation 1.0

La Web Content Accessibility Guidelines 1.0 è diventata W3C Recommendation il 5 Maggio 1999.

Il documento è suddiviso in 14 linee guida, come riportato di seguito:

1. Fornire alternative equivalenti al contenuto audio e video.
2. Non fare affidamento sul solo colore.
3. Usare marcatori e fogli di stile e farlo in modo appropriato.
4. Chiarire l'uso di linguaggi naturali.
5. Creare tabelle che si trasformino in maniera elegante.
6. Assicurarsi che le pagine che danno spazio a nuove tecnologie si trasformino in maniera legante.
7. Assicurarsi che l'utente possa tenere sotto controllo i cambiamenti di contenuto nel corso del tempo.
8. Assicurare l'accessibilità diretta delle interfacce utente incorporate.
9. Progettare per garantire l'indipendenza da dispositivo.
10. Usare soluzioni provvisorie.
11. Usare le tecnologie e le raccomandazioni del W3C.
12. Fornire informazione per la contestualizzazione e l'orientamento.
13. Fornire chiari meccanismi di navigazione.
14. Assicurarsi che i documenti siano chiari e semplici.

Ogni linea guida, a sua volta, comprende uno o più punti di controllo (*checkpoint*) che spiegano come vengono applicate le linee guida in uno specifico contesto. Ad ogni punto di controllo è stato quindi assegnato dal gruppo di lavoro, un valore di priorità che si basa sull'impatto che quel checkpoint ha rispetto alle problematiche di accessibilità.

[Priority 1]

Riguardano le norme che **devono** essere soddisfatte. In caso contrario uno o più gruppi di utenza troveranno impossibile l'accesso alle informazioni di un documento.

[Priority 2]

Riguardano quelle norme che **dovrebbero** essere soddisfatte. In caso contrario uno o più gruppi di utenza troveranno delle difficoltà nell'accesso alle informazioni di un documento.

[Priority 3]

Riguardano quelle norme che **potrebbero** essere soddisfatte. In caso contrario uno o più gruppi di utenza troveranno qualche difficoltà nell'accesso alle informazioni di un documento.

Essere conformi alle priorità 1, 2 o 3, dà la possibilità allo sviluppatore di dichiarare all'interno del proprio sito il livello di conformità adottato:

- **Conformità di Livello "A"**: vengono soddisfatti tutti i checkpoints di Priorità 1;
- **Conformità di Livello "Doppia-A"**: vengono soddisfatti tutti i checkpoints di Priorità 1 e quelli di Priorità 2;
- **Conformità di Livello "Tripla-A"**: vengono soddisfatti i checkpoints di tutte le Priorità 1, 2 e 3.

La dichiarazione di conformità a uno dei tre livelli può essere fatta secondo uno dei criteri suggeriti nella stessa raccomandazione[49], il più diffuso dei quali è l'icona creata dallo stesso Consorzio (vedi § 13.4).

13.2.2 La WCAG Recommendation 2.0

La nuova versione 2.0 delle WCAG fa ancora parte di un documento in via di revisione che secondo quanto dichiarato dallo stesso W3C dovrebbe diventare Raccomandazione alla fine del 2005.

Le WCAG 2.0 sono organizzare secondo quattro principi di progettazione per l'accessibilià del Web:

1. Il contenuto deve essere percepibile. (Content must be **perceivable)**

2. Gli elementi dell'interfaccia del contenuto devono essere funzionali. (Interface elements in the content must be **operable)**

3. Il contenuto e i controlli devono essere comprensibili. (Content and controls must be **understandable)**

[49] cfr. Documento all'indirizzo http://www.w3.org/TR/WCAG10/#Conformance

4. Il contenuto deve essere abbastanza robusto da poter lavorare con tecnologie web attuali e future. (Content must be **robust** enough to work with current and future Web technologies)

Ad ogni principio sono associate delle linee guida che definiscono come I principi vengono applicati in una specifica area. Sotto ciascuna linea guida ci sono i criteri di successo, le definizioni, i vantaggi e gli esempi. I criteri di successo possono essere di livello 1, 2 o 3. le basi per determinare la conformità alle WCAG 2.0 Working Draft sono propio i criteri di successo.

Principle 1: Content must be perceivable.

Guideline 1.1: Provide text alternatives for all non-text content.

Guideline 1.2: Provide synchronized alternatives for multimedia.

Guideline 1.3: Ensure that information, functionality, and structure can be separated from presentation.

Guideline 1.4: Make it easy to distinguish foreground information from background images or sounds.

Principle 2: Interface elements in the content must be operable.

Guideline 2.1: Make all functionality operable via a keyboard interface.

Guideline 2.2: Allow users to control time limits on their reading or interaction.

Guideline 2.3: Allow users to avoid content that could cause seizures due to photosensitivity.

Guideline 2.4: Provide mechanisms to help users find content, orient themselves within it, and navigate through it.

Principle 3: Content and controls must be understandable.

Guideline 3.1: Make text content readable and understandable.

Guideline 3.2: Make the placement and functionality of content predictable.

Principle 4: Content must be robust enough to work with current and future technologies.

Guideline 4.1: Use technologies according to specification.

Guideline 4.2: Ensure that user interfaces are accessible or provide an accessible alternative(s)

Tabella 1 – I quattro principi e le relative linee guida come riportati nel documento draft delle WCAG 2.0 all'indirizzo http://www.w3.org/TR/2005/WD-WCAG20-20050630/

13.2.3 Differenze fra WCAG 1.0 e WCAG 2.0

Le WGCAG 2.0 Working Draft sono molto più robuste delle WCAG 1.0, indipendenti dalla tecnologia, testabili ed hanno molte più informazioni di supporto. Come già accennato la versione 2.0 è ancora in fase di revisione, quindi si deve continuare a fare riferimento alla 1.0.

Per maggiori dettagli sulle differenze tra le due versioni si consiglia di visitare le pagine ufficiali del WAI all'indirizzo http://www.w3.org/WAI/intro/wcag.php.

13.3 Legislazione di riferimento

Come ci ricorda Charles McCathieNevile[50], *"per quasi due anni esperti di diversi paesi e di diversi campi hanno messo a punto le linee guida, lavorando con gli utenti, gli sviluppatori e i progettisti."* Dalla loro pubblicazione nel 1999, *"le linee guida sono state riconosciute come standard di best practice dai governi, compresi il Parlamento Europeo e i suoi stati membri, il Canada, l'Australia e i suoi stati, diversi stati degli Stati Uniti e da organizzazioni industriali."*

A partire dalla WCAG 1.0, molti governi hanno iniziato ad emanare leggi in porposito di accessibilità dei siti pubblici, ma negli ultimi anni sembra che il problema abbia interessato un numero maggiore di paesi, compresa l'Italia, dove la situazione era ferma da anni a proposte di legge o semplici circolari.

13.3.1 La Rehabilitation Act Amendments (Section 508)

Il 7 agosto 1998 il Presidente Clinton firma la legge denominata Rehabilitation Act Amendments che copre l'accesso ai servizi e ai fondi federali. La legge rafforza e migliora la Section 508 della Rehabilitation Act del 1973. con l'uscita della legge, tutte le agenzie federali che sviluppano, mantengono o usano tecnologia elettronica per l'informazione, devono assicurare che ogni prodotto sia accessibile a tutti gli impiegati e agli utenti con disabilità. La Section 508 fa riferimento a tutte le pagine Internet federali e non viene ancora applicata all'industria privata. Secondo la Section 508 tutti i siti federali dovevano adeguarsi ai requisiti di accessibilità entro il 21 giugno del 2001, pena l'esclusione dai finanziamenti dati dal fondo federale e la concreta possibilità di essere citati dagli utenti con disabilità.

13.3.2 La Commissione Europea

Anche in Europa ci si muove in questo senso già dal 1999, hanno in cui il Consiglio pubblica il documento "e-Europe – An Information Society for all"

[50] Charles McCathieNevile è membro del WAI. La dichiarazione riportata nel testo è tratta da una intervista che lo stesso McCathieNevile ha rilasciato per la rivista "I quaderni di Telèma" della Fondazione Ugo Bordoni, numero speciale dedicato all'accessibilità e disponibile online all'indirizzo http://www.fub.it/attivita_telema_lista.php?anno_c=2002

in cui viene richiesto a tutti gli stati membri di rendere accessibili struttura e informazione di tutti i siti di pubblica utilità, entro il 2001.

Il 2003 è stato l'*Anno europeo delle persone con disabilità*. Durante tutto l'arco si sono succedute molte manifestazioni, eventi, incontri, concorsi, progetti, in tutti gli stati membri dell'Unione, che dovevano principalmente servire a sensibilizzare i cittadini sulle tematiche in gioco. A dicembre dello stesso anno si *"si è svolta a Roma la Conferenza di chiusura dell'Anno europeo delle persone con disabilità. La manifestazione, cui hanno partecipato 500 delegati provenienti da tutti i paesi dell'Unione Europea e dei paesi aderenti, ha esaminato i primi risultati dell'Anno Europeo e lancia le strategie future nel quadro dell'Europa allargata.*[51]"

Sul portale dell'Unione Europea è possibile inoltre visionare e scaricare una serie di documenti sulle problematiche legate a Internet e all'accessibilità dei siti, ormai all'attenzione del Consiglio da diversi anni, tra cui:

- la Società dell'Informazione (http://europa.eu.int/scadplus/leg/it/s21012.htm)
- Europa, politica per l'accessibilità del web (http://europa.eu.int/geninfo/accessibility_policy_it.htm)
- 2010: un'Europa accessibile per tutti – Relazione del gruppo di esperti istituito dalla Commissione Europea – ottobre 2003 (http://europa.eu.int/comm/employment_social/index/final_report_ega_it.pdf)
- e-Europe 2005 http://europa.eu.int/information_society/eeurope/2005/index_en.htm
- eEurope 2002: accessibilità e contenuto dei siti Internet delle amministrazioni pubbliche - Comunicazione della Commissione al Consiglio, al Parlamento Europeo, al Comitato Economico e Sociale e al Comitato delle Regioni

13.3.3 La situazione italiana

La tabella riporta l'elenco di alcuni dei molti progetti di legge che sono stati proposti al Parlamento Italiano sin dal 2001. L'unico che ha concluso l'iter e il progetto di legge n. 3978 del 15.05.2003, diventato Legge Stanca il 9.01.2004.

[51] per un approfondimento su quanto realizzato durante l'anno, vedere il sito italiano all'indirizzo http://www.annoeuropeodisabili.it/default.html, da cui si può accedere al sito ufficiale europeo.

PdL.	titolo	anno	proponenti	# articoli
3978	Disposizioni per favorire l'accesso dei soggetti disabili agli strumenti informatici	15.05.2003	Stanca	10
3713	Disposizioni in materia di diritto di accesso ai servizi e alle risorse informatiche pubbliche per I cittadini diversamente abili	25.02.2003	Labate	12
3846	Disposizioni per favorire il diritto di accesso ai servizi informatici e telematici pubblici da parte dei cittadini	01.04.2003	Di Teodoro	7
3486	Norme per il diritto di accesso ai servizi e alle risorse telematiche pubbliche e di pubblica utilità da parte dei cittadini diversamente abili	16.12.2002	Campa-Palmieri	8
2950	Disposizioni per consentire l'accesso ai siti internet della pubblica amministrazione ai soggetti portatori di handicap	02.07.2002	Jannone	1
494	Disposizioni per consentire l'accesso ad Internet ai soggetti portatori di handicap	05.06.2001	Bono	4 (uguale a 232)
232	Disposizioni per consentire l'accesso ad Internet ai soggetti portatori di handicap	30.05.2001	Piscitello	4

Fino all'approvazione della Legge Stanca in Italia erano state diffuse solo circolari, che, in quanto tali, avevano esclusivamente il merito di informare ma non di imporre nessuna regola:

- Circolare del CNR n.25/2000 (Consiglio Nazionale delle Ricerche – Direzione Generale – Roma, 16 novembre 2000) – **Piano di comunicazione e linee di indirizzo per la produzione dell'informazione su tutti i siti del dominio cnr.it**
- Circolare 13 marzo 2001 (Circolare Funzione Pubblica) – **Linee guida per l'organizzazione, l'usabilità e l'accessibilità dei siti web delle pubbliche amministrazioni**
- Circolare AIPA del 6 settembre 2001 (AIPA/CR/32) - **Criteri e strumenti per migliorare l'accessibilità dei siti web e delle applicazioni informatiche a persone disabili**
- Direttiva del Presidente del Consiglio dei Ministri del 30 maggio 2002 – **L'accessibilità dei siti web**

13.3.4 La legge Stanca

La legge è uscita il 9 gennaio 2004, con il titolo **"Disposizioni per favorire l'accesso dei soggetti disabili agli strumenti informatici"**. È meglio conosciuta come Legge Stanca dal nome del Ministro per l'Innovazione e le Tecnologie, che l'ha promossa.

La legge è composta da 12 articoli e prevede (vedi art. 10) l'uscita di un regolamento di attuazione in cui vengono definiti:

a) i criteri e i princìpi operativi e organizzativi generali per l'accessibilità;

b) i contenuti di cui all'articolo 6, comma 2;

c) i controlli esercitabili sugli operatori privati che hanno reso nota l'accessibilità dei propri siti e delle proprie applicazioni informatiche;

d) i controlli esercitabili sui soggetti di cui all'articolo 3, comma 1.

In effetti il regolamento di attuazione sarebbe dovuto uscire *"entro novanta giorni dalla data di entrata in vigore della presente legge"*, come si legge nel testo stesso della Legge Stanca, mentre sono usciti il 1 marzo 2005[52].

L'8 luglio del 2005 sono stati pubblicati come Decreto Ministeriale anche i **"Requisiti tecnici e i diversi livelli per l'accessibilità agli strumenti informatici"**. I documenti sono composti dal testo stesso del decreto e 6 allegati specifici, come riportato nell'elenco sotto:

1. **Testo del Decreto**
2. **Allegato A**: Verifica tecnica e requisiti di accessibilità delle applicazioni basate su tecnologie internet.
3. **Allegato B**: Metodologia e criteri di valutazione per la verifica soggettiva dell'accessibilità delle applicazioni basate su tecnologie internet.

[52] Il testo completo della legge Stanca e dei regolamenti sono reperibili all'indirizzo http://www.pubbliaccesso.gov.it/normative/regolamento.htm

4. **Allegato C**: Requisiti tecnici di accessibilità per i personal computer di tipo desktop e portatili.
5. **Allegato D**: Requisiti tecnici di accessibilità per l'ambiente operativo, le applicazioni e i prodotti a scaffale.
6. **Allegato E**: Logo di accessibilità dei siti Web e delle applicazioni realizzate con tecnologie Internet.
7. **Allegato F**: Importi massimi dovuti dai soggetti privati come corrispettivo per l'attività svolta dai valutatori.

13.4 Valutare l'accessibilità

Ci sono diversi metodi per valutare l'accessibilità di un sito web. È vero che un singolo approccio non può evidenziare tutti i problemi riscontrabili, ma è bene adottare più metodologie insieme.

Dopo aver scelto un campione esemplificativo delle pagine del sito da testare:

- esaminarle con più browser grafici, come Internet Explorer, Opera, Mozilla, Netscape Navigator, possibilmente in diverse versioni
- esaminare le pagine con browser vocali e di solo testo
- usare almeno due programmi generali per la valutazione automatica dell'accessibilità e analizzare i dati ottenuti
- eseguire un controllo manuale dell'accessibilità secondo i checkpoints del WCAG 1.0 http://www.w3.org/TR/WCAG10/full-checklist.html
- eseguire un test di usabilità sulle problematiche dell'accessibilità, possibilmente includendo anche utenti diversamente abili
- riassumere tutti i problemi riscontrati e cercare di risolverli

13.4.1 I validatori automatici

Sul sito del WAI è possible scegliere fra diversi validatori generali automatici o semiautomatici[53]. Come esempio riportiamo di seguito alcuni dei tool maggiormente utilizzati.

Bobby http://webxact.watchfire.com/

Bobby è un tool sviluppato dalla Watchfire. Non esistono versioni scaricabili del programma, ma ci sono diverse opzioni, suggerite dagli stessi sviluppatori. Una di queste è WebXACT, un servizio gratuito online che permette di valutare i problemi di accessibilità, qualità e sicurezza del contenuto delle singole pagine web.

[53] http://www.w3.org/WAI/ER/existingtools.html#General

Figure 23 – Home page di WebXACT

Cynthia Says http://www.cynthiasays.com/

Il tool è stato progettato per individuare errori relative alla Section 508 e alle Linee Guida del W3C WCAG 1.0. Valida una pagina alla volta ed oltre alla versione online, ne esiste anche una scaricabile. Uno degli scopi del portale di Cynthia è anche quello di educare gli sviluppatori di siti web riguardo alle problematiche dell'accessibilità[54].

Figure 24 – Cynthiasays.com

Torquemada http://www.webxtutti.it/testa.htm

L'unico validatore in lingua italiana è stato messo a punto da un gruppo di ricercatori della Fondazione Ugo Bordoni di Roma, e si chiama Torquemada. La Fondazione Ugo Bordoni (FUB) organizza da anni seminari sull'accessibilità ed ha ultimamente messo in linea un sito su queste

[54] Vedere le risorse all'indirizzo http://www.hisoftware.com/acccenter/accessibility.html

tematiche con molte informazioni utili, link e il tool per la validazione delle pagine.

Figure 25 – Torquemada: unico validatore italiano (ora anche in lingua inglese)

13.4.2 Dichiarazione di conformità

Una volta passati i test di valutazione dell'accessibilità è possibile dichiarare che una pagina o un sito abbiano raggiunto un livello di conformità "A", "Doppia-A" o "Tripla-A". Ciò indica infatti che la pagina in questione rispetta tutti i punti di controllo definiti dalle WCAG, rispettivamente di Priorità 1, 2 o 3.

Il W3C mette a disposizione tre diverse icone relative a ciascun livello di conformità, che chiunque può utilizzare all'interno del proprio sito. Riportiamo di seguito le tre diverse icone con il codice relativo.

[Livello A]

```
<a href="http://www.w3.org/WAI/WCAG1A-Conformance"
      title="Explanation of Level A Conformance">
  <img height="32" width="88"
         src="http://www.w3.org/WAI/wcag1A"
         alt="Level A conformance icon,
         W3C-WAI Web Content Accessibility Guidelines 1.0"></a>
```

[Livello Doppia-A]

```
<a href="http://www.w3.org/WAI/WCAG1AA-Conformance"
      title="Explanation of Level Double-A Conformance">
  <img height="32" width="88"
         src="http://www.w3.org/WAI/wcag1AA"
         alt="Level Double-A conformance icon,
         W3C-WAI Web Content Accessibility Guidelines 1.0"></a>
```

[Livello Tripla-A]

```
<a href="http://www.w3.org/WAI/WCAG1AAA-Conformance"
      title="Explanation of Level Triple-A Conformance">
  <img height="32" width="88"
         src="http://www.w3.org/WAI/wcag1AAA"
         alt="Level Triple-A conformance icon,
         W3C-WAI Web Content Accessibility Guidelines 1.0"></a>
```

Interagire con il web

14.1 Interagire col web

Le interazioni col web sono una parte molto importante per affrontare la progettazione e la realizzazione di un sito web. In questa sezione prendiamo in analisi questi argomenti:

- l'invio di informazioni al server;
- le form e gli elementi contenuti;
- l'interazione con le pagine web;
- alcuni eventi in dettaglio.

14.1.1 Interazioni col web: le form

Il web di oggi è molto spesso usato anche come mezzo per realizzare applicazioni che richiedono una interazione con l'utente che va oltre la normale navigazione. Un tipico esempio è la raccolta di informazioni dai visitatori del sito. Vediamo in dettaglio alcune parti essenziali.

In prima istanza scriviamo il codice necessario a far si che si possa inviare una email (per mezzo del client di posta normalmente presente sui nostri computer). Di solito questa tecnica molto semplice che non richiede nessun programma dal lato server, è usata per far inviare ad un indirizzo prefissato dei commenti o delle segnalazioni via email. Come? Per mezzo degli elementi `form` e `input`.

L'esempio mostra direttamente la sintassi tipica usata con le form:

```
<form
 action="mailto:xxx.yyy@iit.cnr.it"
 method="post" enctype="text/plain">
 <input type="text" name="subject"
  value="ciao" />
 <input type="submit" name="empty"
  value="invia" />
</form>
```

Con sistemi più complessi, cioè applicazioni che dal lato server generano dinamicamente le pagine è possibile far si che il contenuto, e quindi anche le form, siano dipendenti dai dati immessi e dalle interazioni dell'utente. In particolare l'indirizzo della pagina e gli altri parametri, come i valori degli elementi input delle form, possono influenzare il comportamento comportamento dell'applicativo che sta sul server e produce le pagine HTML.

I CGI sono tra gli applicativi più vecchi che ancora si utilizzano sul web per generare pagine Web dinamiche.

14.1.2 Action

L'attributo `action` dell'elemento `form` ha come valore una URI che determina l'azione della form stessa:

1. nel caso lo schema della URI sia `mailto` viene aperto il client per l'invio di una email all'indirizzo indicato;
3. nel caso lo schema della URI sia HTTP al momento del `submit` i dati sono inviati al server e quel valore indica la URI di destinazione.

Ecco un esempio di codice di una `form` con action di tipo `mailto:`

```
<form action="mailto:xxx.yyy@iit.cnr.it"
   method="post" enctype="text/plain">
  …
</form>
```

14.1.3 Method

L'attributo `method` dell'elemento `form` corrisponde al metodo HTTP che sarà usato al momento dell'invio della `form` stessa per mezzo della fase di *submit*. Il suo valore indica in che modo le informazioni vengono inviate al server e può assumere due diversi valori:

- se il valore è `post` la trasmissione dei dati della form è costituita da due parti:
  - il browser (client) comunica al server che ci sarà una `post` all'indirizzo indicato nella `action`;
  - in una trasmissione apposita il client trasmette al server i dati della form.
- se il valore è `get` la trasmissione dei dati della form al server viene effettuata in un singolo passo.

Come scegliere tra `get` o `post`?

Da quanto detto si deduce che il metodo `get` è da preferire nel caso sia importante minimizzare il numero di trasmissioni. Comunque il metodo metodo `get` ha alcuni limiti che sono nelle dimensioni e di codifica dei parametri da trasmettere. Inoltre usando il metodo `get` i parametri sono immessi alla fine della URI ed è meno sicuro il suo uso rispetto alla `post`, anche se è solo una questione di facciata (la sicurezza si affronta passando su HTTPS). Infine il metodo `post` è indicato per le form complesse ed è necessario in quelle form che possono contenere file da trasmettere al server.

Quando il browser chiede una pagina web al server via HTTP, la richiesta viene effettuata per mezzo di una `get`. Quindi l'uso di una `form` contenente il metodo `get` è equivalente del punto di vista del protocollo HTTP ad una normale richiesta di pagina (senza form).

14.1.4 Enctype

L'attributo `enctype` dell'elemento `form` serve per indicare il formato di codifica da usare per trasmettere i dati della form verso il server. I possibili valori che può assumere:

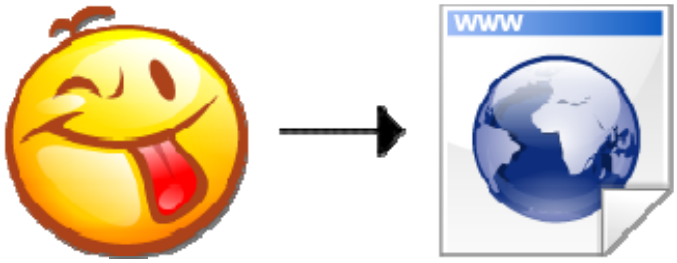
- `application/x-www-form-urlencoded`
  - è il valore predefinito. Questa tipo di codifica compone una stringa convertendo tutti i campi della form in una sequenza di associazioni `nome-campo=valore`;
- `text/plain`
  - è da usarsi con le *form* che hanno una `action` che ha schema `mailto`;
- `multipart/form-data`
  - è da usarsi soltanto con il metodo `post`. Questa codifica compone ogni campo della form in una o più parti di un documento di tipo MIME.

Il **MIME** (Multipurpose Internet Mail Extensions) indica una estensione del formato delle e-mail per permettere la codifica di messaggi con caratteri diversi dall'ASCII, di messaggi non testuali, di messaggi con diverse tipologie di contenuto (immagini, testo, ...).

Le RFC dalla *2045* alla *2049* sono il riferimento per il MIME, mentre nel protocollo HTTP è indicato il comportamento dei client e server con le post di tipo `multipart`.

14.1.5 Interazioni con le form via HTTP

L'interazione tipica di un utente con le pagine contenenti le form è la seguente:



L'utente riempie gli elementi (`input`, ...) nelle form sul browser (tipicamente le caselle di testo).

L'utente seleziona nella form uno degli elementi `input` con `type="submit"`, cioè fa click su un pulsante nella pagina.

I dati vengono inviati dal client al server, usando metodo e codifica indicati negli attributi della form dopo che il browser li ha codificati appositamente.

Il server elabora i dati inviati dal client e restituisce una risposta, che normalmente è una pagina HTML.

14.2 Gli elementi delle form

14.2.1 L'elemento input

L'elemento `input` si usa per definire gli elementi di controllo più diffusi nelle form.

Gli attributi più comuni sono:

`name:` indica il nome del campo, viene usato al momento dell'invio della form;
`value:` indica il valore predefinito del campo;
`type:` indica il tipo del campo. Vediamo quali valori può assumere.

Di seguito sono riportati i possibili valori possibili per l'attributo `type` del tag `input`:

`submit`: pulsanti per inviare i dati
`file`: campo per immettere file
`button`: pulsante generico
`hidden`: campo nascosto
`image`: campo immagine
`password`: campo utile per dati riservati
`reset`: campo per azzerare la form ai valori predefiniti
`radio`: campo di scelta di uno tra più elementi
`checkbox`: campo per la scelta multipla.

Affinché l'elemento `input` sia correttamente interpretato gli attributi `type` e `name` sono necessari.
I tipi di input più comuni sono il text e il submit:

- Usando l'attributo `type="text"` in un elemento `input` si crea un campo che permette l'inserimento di una riga di testo. La larghezza dei campi di questo tipo, come di altri, è impostabile con l'attributo `size`.

Usando `type="submit"` si crea un campo che, quando attivato, serve a inviare i dati della form.

Ecco un esempio di input text e submit:

```
<input type="text"
  name="testo" value="123"
  size="4" />
<input type="submit"
  name="empty"
  value="invia" />
```

Usando l'elemento `input` con il `type="password"` si ha la possibilità di creare un campo che permette l'inserimento di dati riservati. Questo **non ha implicazioni con la sicurezza del trasferimento** dei dati, che è comunque identica per tutti i campi della form. È invece solo una protezione sullo schermo che il browser può attuare, di solito sostituendo ogni carattere digitato con un simbolo *.

Con l'elemento `input` e il `type="hidden"` si chiede al browser che il campo in questione non sia presentato all'utente, ma che comunque sia inviato nella trasmissione al server al momento dell'invio della `form`. Con questo tipo generalmente si introducono dei valori predefiniti della form da inviare durante la fase di *submit*.

Con l'elemento `input` e il `type="file"` si crea un campo che permette la selezione di un file sul client che verrà trasmesso al server al momento dell'invio della `form`.

Ecco alcuni esempi riguardanti ancora il campo `input:`

```
<input type="password"
  name="riservato" size="4" />
<input type="hidden"
  name="nascosto"
  value="CorsoHTML" />
<input type="file"
  name="immagine" />
```

(Ovviamente il campo hidden non è visibile!)

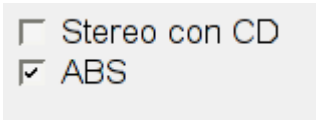
In congiunzione con il `type="file"` si può usare l'attributo `accept` per indicare il Content-Type dei file accettati (per esempio: `accept="image/*"` per accettare tutte le immagini).

L'elemento `input` e il `type="checkbox"` crea una selezione multipla, composta da tutti gli `input` che hanno l'attributo `name` con lo stesso valore.

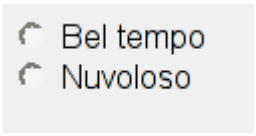
L'elemento `input` e il `type="radio"` crea una selezione di un valore tra più possibilità, in modo esclusivo, tra tutti gli input della `form` che hanno l'attributo `name` con lo stesso valore.

Ecco alcuni esempi riassuntivi di questi due tipi:

```
<input type="checkbox"
   name="accessori"
   value="stereo" />
Stereo con CD
<input type="checkbox"
   checked="checked"
   name="accessori"
   value="ABS" />
ABS
```



```
<input type="radio"
   name="previsioni"
   value="sole" />
Bel tempo
<input type="radio"
   name="previsioni"
   value="nubi" />
Nuvoloso
```



L'attributo `checked="checked"` del tag `input` si usa per determinare quali degli elementi delle selezioni multiple devono essere attivi in modo predefinito.

14.2.2 L'elemento textarea

Per ottenere aree di una form che possono contenere più righe di testo da inviare, si usa l'elemento `textarea`, per il quale sono disponibili alcuni attributi supplementari:

- `rows`: indica le righe visualizzate dal browser;
- `cols`: indica le colonne visualizzate dal browser;
- `wrap`: determina il comportamento del browser rispetto agli a-capo digitati dall'utente e quelli inviati al server:
  - off, col quale non le righe del testo non sono spezzate dal browser;
  - virtual, con il quale le righe del testo sono spezzate all'occorrenza dal browser, ma senza inserire degli a-capo nel campo trasmesso;
  - physical, gli a-capo sono trasmessi così come sono visti dall'utente.

Ecco un esempio di una semplice `textarea`:

```
<textarea name="testo"
   rows="5" cols="10" />
testo
su più
righe
</textarea>
```

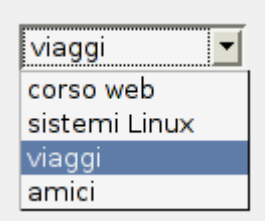
14.2.3 L'elemento select

L'elemento `select` si usa in congiunzione con l'elemento `option` per creare dei campi in una form che permettono di scegliere da una lista di valori. Il

valore selezionato in modo predefinito si indica sul tag **option** con l'attributo **selected="selected"**.

Ecco un esempio di una **select**:

```
<select name="subject">
 <option value="1">corso web</option>
 <option value="2">sistemi Linux</option>
 <option value="3"
  selected="selected">viaggi</option>
 <option value="4">amici</option>
</select>
```



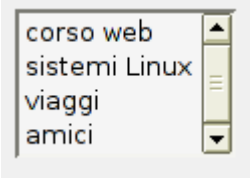
L'elemento **optgroup** si usa per raggruppare gli elementi **option**, in modo da poter organizzare gerarchicamente le varie opzioni. Per indicare una etichetta informativa sul gruppo delimitato da **optgroup** si usa l'attributo **label.**

Ecco un esempio di **select** con l'uso di **optgroup**:

```
<select name="subject">
 <optgroup label="lavoro">
  <option value="1">corso web</option>
  <option value="2">sistemi Linux</option>
 </optgroup>
 <optgroup label="altro">
  <option value="3">viaggi</option>
  <option value="4">amici</option>
 </optgroup>
</select>
```

Usando l'attributo **size** con il tag **select** si indica la dimensione della selezione, il cui valore predefinito è 1, che fa si che nei comuni browser grafici appaia come una lista a tendina. Per valori maggiori di 1 è associata la dimensione del numero di opzioni visualizzate contemporaneamente nella selezione. Questa caratteristica congiuntamente all'attributo **multiple="multiple"** del tag **select** permette di attivare una scelta multipla, così come nell'esempio seguente:

```
<select name="subject"
  size="4" multiple="multiple">
 <option value=”1”>corso web</option>
 <opiotn value=”2”>sistemi linux</option>
 <option value=”3”>viaggi</option>
 <option value=”4”>amici</option>
</select>
```



14.2.4 Come etichettare i campi

Per etichettare i campi delle form esiste l'elemento `label`, che si usa collegandolo ad un campo di interazione utente presente nella form. I modi possibili per legare l'etichetta al campo sono due e sono equivalenti:

- inserire il campo della form (input, select, ...) all'interno dell'elemento label stesso. Ecco un esempio:

```
<label>Nome
  <input type="text" name="ilnome" />
</label>
```

- usare l'attributo `for="id-campo"` nel tag `label`, come nell'esempio seguente:

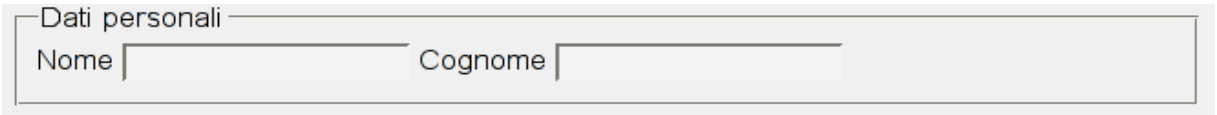
```
<label for="nome">Nome</label>
<input type="text" name="ilnome" id="nome" />
```

14.2.5 Come raggruppare i campi

Per raggruppare i campi di una form per tipologia di dato si può usare l'elemento `fieldset`. L'esempio seguente mostra anche l'uso dell'elemento `legend` da usarsi all'interno dell'elemento `fieldset`, come descrizione della tipologia di campi delimitati da questo gruppo di campi. Ecco un esempio di codice HTML seguito dalla sua visualizzazione tipica, dove si fa uso di entrambi di entrambi questi elementi, nonché di `label`:

```
<fieldset>
  <legend>Dati personali</legend>
  <label>Cognome
    <input type="text" name="cognome" />
  </label>
  <label>Nome
    <input type=”text” name=”nome” />
  </label>
</fieldset>
```

La cui resa in un browser tradizionale è la seguente:



14.3 Presentazione delle form

Gli elementi (`input`, `select`, ...) delle form si possono personalizzare per mezzo dei fogli di stile come gli altri elementi HTML. Le proprietà dei CSS applicabili a questi elementi sono, in larga misura, già viste: `font`, `color`, `margin` ... Inoltre è possibile usare elementi generici come `div` e `span` per identificare e strutturare, nonché presentare gli elementi in questione.

Ecco un esempio di CSS applicato ad una semplice form:

```
input {
  padding: 0.2em;
  background: yellow;
  color: blue;
  border: 1px solid red;
}
```

14.3.1 Alcuni eventi

Per alcune interazioni dell'utente con il browser e per alcune azioni specifiche del browser stesso si generano degli eventi. Molti di questi eventi è possibile intercettarli con alcuni meccanismi messi a disposizione direttamente dai browser. I più comuni ed usati sono:

- click del mouse *(onmouseclick);*
- movimenti del mouse *(onmousemove);*
- caricamento della pagina *(onload);*
- invio di una form *(onsubmit).*

Ad alcuni di questi eventi è possibile associare uno o più comandi, usati di solito per creare un'interazione tra le funzionalità del browser e l'utente.

In questo contesto si sviluppano e si utilizzano i linguaggi di programmazione operanti sui browser: javascript, applet java, ...

14.3.2 Javascript

Javascript è un linguaggio di scripting orientato agli oggetti, che è stato sviluppato inizialmente dalla Netscape Communications con il nome interno di LiveScript. Poco prima del rilascio la sintassi è stata resa simile a quella del linguaggio Java della Sun Microsystems, ecco così che si è cambiata la denominazione in Javascript. Dal 1995 è presente in Navigator 2.0 e dall'anno successivo anche Explorer 3.0 (Jscript), mentre nell'autunno 1996 c'è stata la presentazione di Javascript alla ECMA (European Computer

Manufacturers Association) per la sua standardizzazione con il nome di ECMAScript. Questo comunque non è ancora un nome molto diffuso e si continuano a chiamare Javascript tutte le versioni e varianti di questo linguaggio oggi presenti nei nostri browser.

In questo contesto viene citato molto anche il termine DHTML, ma non ne esiste una definizione precisa. Comunemente con DHTML si intende HTML dinamico, che si modifica o produce direttamente sul client, cioè i browser. Una pagina HTML con javascript può essere inteso come DHTML.

Javascript è un linguaggio di scripting che utilizza alcune funzionalità del browser per interagire con l'utente. Il codice javascript viene inserito all'interno di appositi blocchi, oppure come istruzioni collegate a elementi HTML specifici. Nei browser che supportano javascript queste istruzioni vengono interpretate ed eseguite nel modo opportuno.

Per esempio si possono:

- intercettare molte delle interazioni dell'utente con la pagine Web, e quindi far eseguire compiti specifici;
- far eseguire particolari operazioni al verificarsi di specifiche condizioni come, ad esempio, il caricamento di una pagina o l'invio di una form.

Con la diffusione di javascript è stato introdotto il concetto di DOM, di origine XML. Il DOM è un modello standard che viene usata nei client e nei server web, per accedere, per una eventuale elaborazione, alle strutture che rappresentano l'HTML e gli oggetti correlati.

Con javascript si possono manipolare molti degli attributi nonché la presentazione degli elementi HTML. L'elenco completo degli attributi che è possibile modificare o comunque impostare in ciascun elemento è ancora deducibile dalla DTD. Queste indicazioni sono comunque presenti nell'appendice A.

14.3.2.1 Javascript: dove e perché

Vediamo come si fa ad inserire ed usare codice javascript nelle pagine HTML:

- come codice da eseguire al momento del caricamento della pagina. In questo caso si usa l'elemento <*script*> che racchiude le istruzioni javascript:

```
<script language="Javascript" type="text/javascript">

 … codice javascript …

</script>
```

- come codice da eseguire al verificarsi di una particolare condizione relativa ad un certo elemento:

```
<input type="submit" name="submit" value="Premi questo"
        onclick="…istruzione javascript…" />
```

Uno dei modi più diffusi di JavaScript è quello di creare piccole funzioni da includere nell'HTML, che interagiscono con il browser per effettuare alcuni compiti non possibili con il solo HTML, come ad esempio:

- aprire nuove finestre con dimensioni prefissate;
- controllare direttamente sul client che i valori immessi nelle form rispettino certi criteri;
- creare effetti grafici legati al movimento o ai click del mouse.

Ecco un piccolo esempio di javascript nell'HTML per la validazione del contenuto di una form:

```
<script type="text/javascript" language="javascript">
function validazione(form) {
  return confirm("Sei sicuro di voler inviare questi dati?")
}
</script>

<form action="http://url/della/action"
 method="post" onsubmit="validazione(this)">
 ...
</form>
```

In questo segmento di HTML si introduce una *validazione* nel momento in cui viene inviata la form, cioè con l'evento *onsubmit*. I dettagli di come fare questa validazione sono specifici del linguaggio Javascript e della form e non li prendiamo in considerazione in questo corso.

Ovviamente però vanno fatte tutte le validazioni anche a livello server, quindi l'uso di javascript è confinato a parti non fondamentali della validazione. In pratica va sempre visto come una caratteristica opzionale del browser.

Inoltre è bene indicare la versione di javascript laddove siano usate istruzioni specifiche di versioni diverse, come nell'esempio corrente:

```
<script language="javascript1.2" type="text/javascript">

 … istruzioni javascript (specifiche della versione 1.2) …

</script>
```

È anche opportuno usare l'elemento <*noscript*> affiancandolo a <*script*> in modo da permettere la visualizzazione di codice HTML alternativo in quei browser che non supportano javascript o che comunque non supportano la versione indicata.

14.4 Approfondimenti

14.4.1 Perché seguire gli standard

I motivi sono stati già più o meno visti durante questo corso, vale comunque la pena ricordarli e citarli qui:

- da' la possibilità di vedere le pagine realizzate anche con futuri browser.
- rende portabile su browser specifici (testuali, braille, vocali, ...) il nostro sito e di conseguenza maggiormente fruibile anche da persone disabili.
- sfrutta appieno le possibilità e le funzionalità dei browser.
- rende possibile visualizzare il web sui vari device (palmari, telefoni di ultima generazione, ...).
- dà maggior coerenza nella presentazione delle pagine, anche cambiando browser/client.

14.4.2 Cenni sull'accessibilità

Usando pochi accorgimenti si possono rendere la pagine decisamente più accessibili. In particolare con pochi ritocchi le nostre pagine HTML possono diventare finalmente accessibili, e molto spesso anche più usabili. Ecco in dettagli i punti salienti:

- usare `alt` e `longdesc` sulle immagini, in modo da fornire sia un testo breve alternativo che una lunga descrizione nel caso l'immagine riporti dei dati importanti;
- usare gli elementi in modo corretto per strutturare la pagina, in modo da facilitare l'interpretazione e la facile localizzazione delle parti interessanti;
- usare l'attributo `accesskey` opportunamente per definire il tasto (o comunque la combinazione di tasti) con il quale utilizzare pagine accessibili;
- usare l'attributo `tabindex` sui campi delle form e sui link per definire l'ordine con cui si passa da un elemento all'altro, tipicamente premendo il tasto TAB.

Ecco un esempio di codice HTML con tabindex e accesskey:

```
<a tabindex="1" accesskey="q"
 href="URI-1.html">
 per questo link: <strong>Q<strong>
```

Maggiori informazioni con i riferimenti a motori automatici per una valutazione - **parziale** - dell'accessibilità sono rintracciabili a partire dalla Web Accessibility Initiative (http://www.w3.org/WAI/).

Un valido e utile esempio di motore di valutazione di accessibilità è l'italiano Torquemada http://www.webxtutti.it/testa.htm.

14.4.2.1 HTML e CSS validi

I validatori sono un buono strumento per determinare la correttezza del codice scritto. Eccone un elenco non completo:

- Validatori del W3C (http://www.w3.org/QA/Tools/#validators)
- Validatore per HTML (http://validator.w3.org/)
- Validatore per CSS (http://jigsaw.w3.org/css-validator/)
- Verifica dei link di una pagina (http://validator.w3.org/checklink)

14.4.3 Futuro degli ipertesti su web

Il lavoro attuale del *W3C* sui linguaggi di MarkUp è indirizzato verso una completa confluenza di tutti gli standard verso il formato XML. Anche il lavoro di riorganizzazione dell'HTML verso XHTML è stato rivolto, e lo è tuttora, verso l'XML come lingua universale, non solo del web. Le caratteristiche delle tecnologie web future e in alcuni casi già presenti sono molteplici. Di seguito è riportata una lista di elementi utili per approfondimenti sulle tecnologie promosse dal W3C:

- la modularizzazione (verso XHTML 2.0), rende più semplice l'uso di XHTML per media con caratteristiche e funzionalità limitate – vedi telefoni e altri nuovi media;
- SVG (Scalable Vector Graphics) è un formato per la grafica vettoriale completamente XML che nelle intenzioni potrebbe sostituire Flash con l'uso di altri formati come SMIL (Synchronized Multimedia Integration Language)
- MathML (Mathematical Markup Language) ancora un formato XML permette l'integrazione delle notazioni matematica con l'editoria elettronica sia nel contenuto che nella struttura.
- CSS futuri (3) saranno di supporto per la presentazione anche delle altre tecnologie XML come per esempio SVG.
- Web semantico, con i formati RDF e OWL: si veda il sito del w3c.org
- Servizi e applicazioni distribuite via web: Web Services e SOAP.

14.4.4 Metadati e indicizzazione

I metadati sono informazioni supplementari che non vengono presentate direttamente nel client/browser, ma che possono essere d'aiuto all'interpretazione della pagina HTML, sia dai motori di ricerca, ma anche dai browser stessi.

Il tag `<meta>` si usa per inserire *metadati* all'interno del codice HTML.

Esempi dei metadati più diffusi:

- <meta name="description" content="Breve descrizione della pagina HTML" />
- <meta name="keywords" content="web, html, css, suggerimenti, tutorial" />

Alcune di queste informazioni (keywords, description, ...) vengono utilizzate per l'indicizzazione da parte dei motori di ricerca. È quindi buona norma inserire *keywords* e *description*.

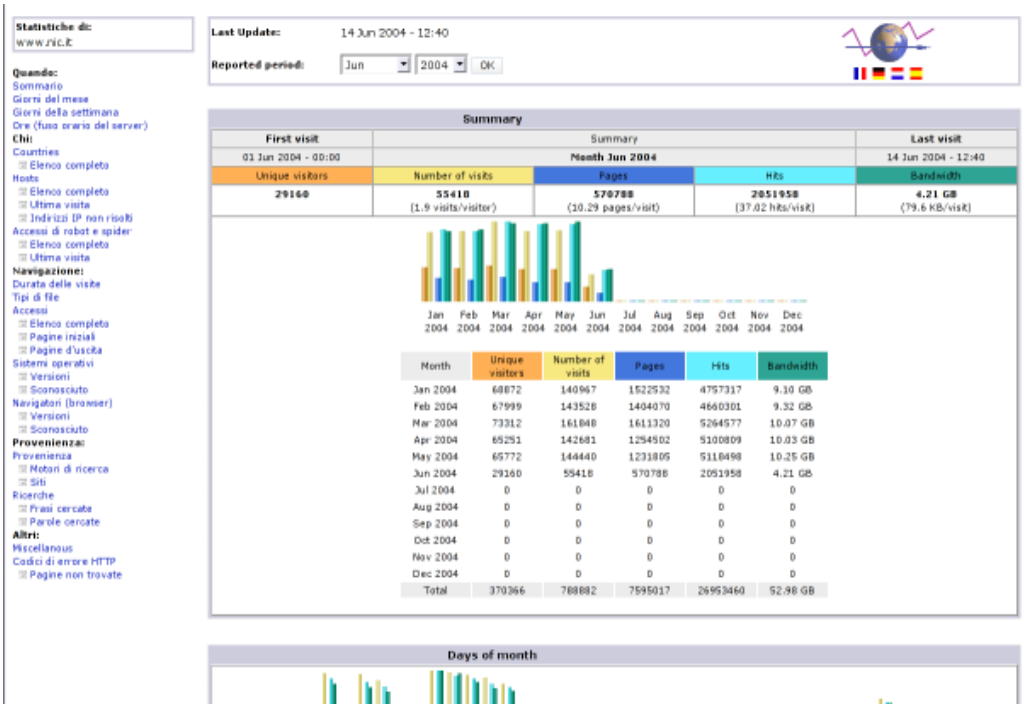
Nota: un suggerimento per l'indicizzazione (google-like) tra i molti possibili è quello di fare collegamenti tra le pagine e la risorsa primaria.

14.4.5 Statistiche del sito

Una parte importante della messa online di un sito web è l'analisi dei log del sito, cioè delle registrazioni che il server web ha fatto di ciascuna pagina visitata. Le informazioni raccolte dai server e scritte nei log sono molteplici e vanno dalla data e l'ora dell'accesso, alla dimensione della pagina scaricata, dal tipo di browser usato alla pagina di provienza...

Le statistiche fatte a posteriori su questi log posso essere molto utili per una serie di fattori diversi:

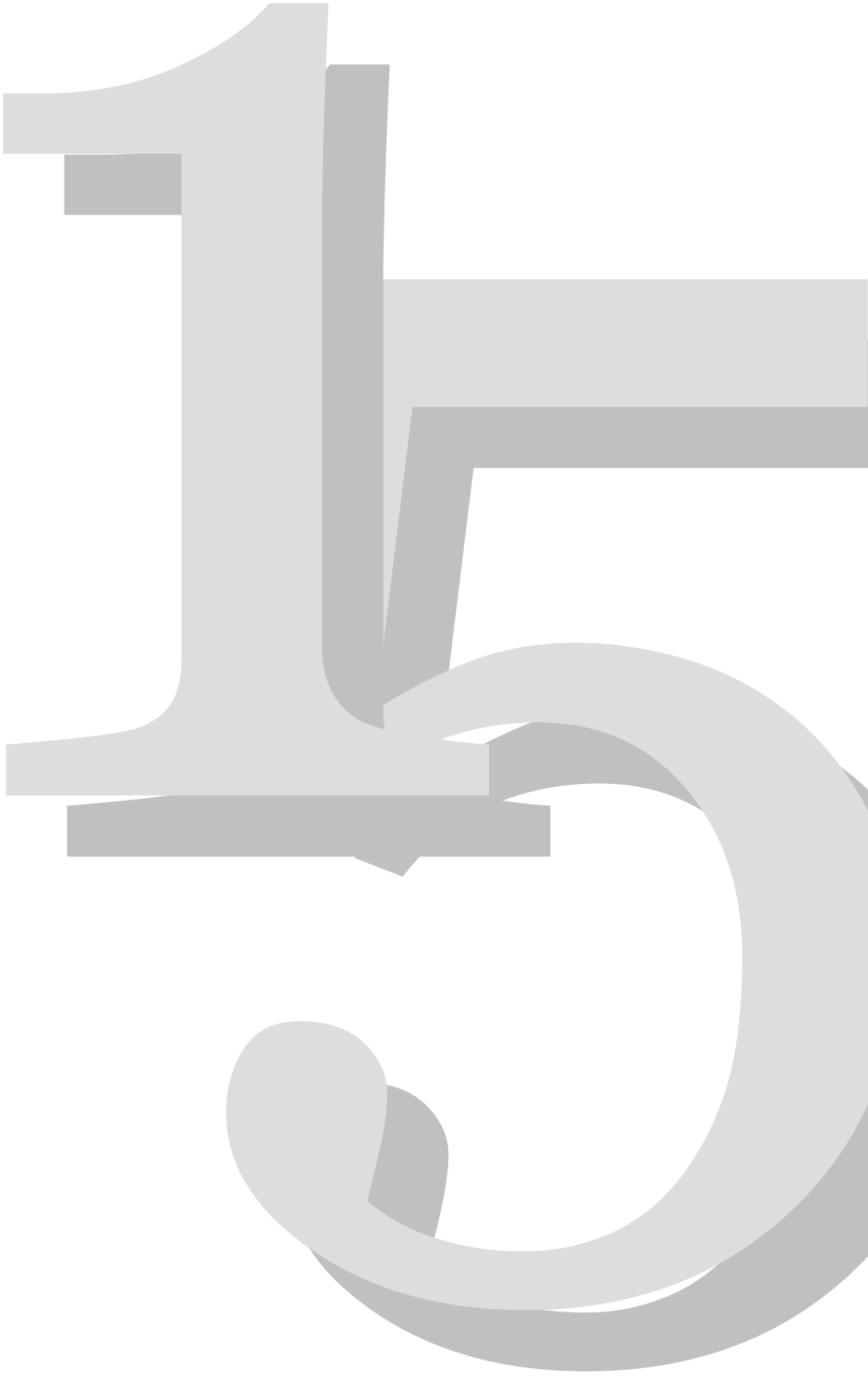
- avere informazioni immediate e potenti riguardanti tutto il sito, con il dettaglio di ciascuna pagina HTML;
- avere un'idea di quanti e quali (gli orari, la velocità di lettura...) utenti frequentano il sito;
- sapere quali sezioni del sito vengono visitate di più (o di meno!);
- sapere quali frasi sono usate sui motori di ricerca o da che siti si accede a questo sito.



pagina di statistiche di log generata dal software libero awstats (http://awstats.sf.net/)

L'analisi dei log è utile anche per motivi di sicurezza. Esistono strumenti - anche software libero - che elaborando i log forniscono statistiche estremamente dettagliate da cui è possibile dedurre molti dati sia sull'utenza che sul traffico generato.

Appendici



Albero degli elementi HTML fino al terzo livello

```
html
 |_(head,
 |   |_((script |
 |   |    |_(#PCDATA)
 |   |
 |   |___style |
 |   |    |_(#PCDATA)
 |   |
 |   |___meta |
 |   |    |_EMPTY
 |   |
 |   |___link |
 |   |    |_EMPTY
 |   |
 |   |___object)*,
 |   |    |_(#PCDATA |
 |   |    |__h1 | ...
 |   |    |__h2 | ...
 |   |    |__h3 | ...
 |   |    |__h4 | ...
 |   |    |__h5 | ...
 |   |    |__h6 | ...
 |   |    |__ul | ...
 |   |    |__ol | ...
 |   |    |__dl | ...
 |   |    |__p | ...
 |   |    |__div | ...
 |   |    |__pre | ...
 |   |    |__blockquote | ...
 |   |    |__address | ...
 |   |    |__hr | ...
 |   |    |__table | ...
 |   |    |__form | ...
 |   |    |__fieldset | ...
 |   |    |__br |
 |   |    |   |_EMPTY
 |   |    |
 |   |    |__span |....
 |   |    |
 |   |    |__em |....
 |   |    |
 |   |    |__strong |....
 |   |    |
 |   |    |__dfn |....
```

```
|   |   |
|   |   |__code |....
|   |   |
|   |   |__samp |....
|   |   |
|   |   |__kbd |....
|   |   |
|   |   |__var |....
|   |   |
|   |   |__cite |....
|   |   |
|   |   |__abbr |....
|   |   |
|   |   |__acronym |....
|   |   |
|   |   |__q |....
|   |   |
|   |   |__tt |....
|   |   |
|   |   |__i |....
|   |   |
|   |   |__b |....
|   |   |
|   |   |__big |....
|   |   |
|   |   |__small |....
|   |   |
|   |   |__sub |....
|   |   |
|   |   |__sup |....
|   |   |
|   |   |__bdo |....
|   |   |
|   |   |__a |....
|   |   |
|   |   |__img |
|   |   |   |_EMPTY
|   |   |
|   |   |__map |....
|   |   |
|   |   |__object | ...
|   |   |__input |
|   |   |   |_EMPTY
|   |   |
|   |   |__select |....
|   |   |
|   |   |__textarea |
|   |   |   |_(#PCDATA)
|   |   |
|   |   |__label |....
|   |   |
```

```
|   |    |__button |....
|   |    |
|   |    |__ruby |.....
|   |    |
|   |    |__ins | ...
|   |    |__del | ...
|   |    |__script | ...
|   |    |__noscript | ...
|   |    |__param)*
|   |       |_EMPTY
|   |
|   |
|   |__((title,
|   |     |_(#PCDATA)
|   |
|   |____(script | ...
|   |_____style | ...
|   |_____meta | ...
|   |_____link | ...
|   |_____object)*, ...
|   |____(base,
|   |      |_EMPTY
|   |
|   |_____(script | ...
|   |______style | ...
|   |______meta | ...
|   |______link | ...
|   |______object)*)?) | ...
|   |___(base, ...
|   |____(script | ...
|   |_____style | ...
|   |_____meta | ...
|   |_____link | ...
|   |_____object)*, ...
|   |____(title, ...
|   |_____(script | ...
|   |______style | ...
|   |______meta | ...
|   |______link | ...
|   |______object)*)))) ...
|
|__body)
    |_(h1 |....
    |
    |__h2 |....
    |
    |__h3 |....
    |
    |__h4 |....
    |
    |__h5 |....
```

|
|__h6 |....
|
|__ul |....
|
|__ol |....
|
|__dl |....
|
|__p |....
|
|__div |....
|
|__pre |....
|
|__blockquote |....
|
|__address |....
|
|__hr |
| |_EMPTY
|
|__table |....
|
|__form |....
|
|__fieldset |....
|
|__ins |....
|
|__del |....
|
|__script | ...
|__noscript)+ ....

15.1.1 Sottoalberi degli elementi

Di seguito sono riportati i sottoalberi degli elementi più utilizzati.

15.1.1.1 SPAN

```
|__span
   |_(#PCDATA |
   |__br | ...
   |__span | ...
   |__em | ...
   |__strong | ...
   |__dfn | ...
   |__code | ...
   |__samp | ...
   |__kbd | ...
   |__var | ...
   |__cite | ...
   |__abbr | ...
   |__acronym | ...
   |__q | ...
   |__tt | ...
   |__i | ...
   |__b | ...
   |__big | ...
   |__small | ...
   |__sub | ...
   |__sup | ...
   |__bdo | ...
   |__a | ...
   |__img | ...
   |__map | ...
   |__object | ...
   |__input | ...
   |__select | ...
   |__textarea | ...
   |__label | ...
   |__button | ...
   |__ruby | ...
   |__ins | ...
   |__del | ...
   |__script | ...
   |__noscript)* ...
```

15.1.1.2 EM

|__em
  |_(#PCDATA |
  |__br | ...
  |__span | ...
  |__em | ...
  |__strong | ...
  |__dfn | ...
  |__code | ...
  |__samp | ...
  |__kbd | ...
  |__var | ...
  |__cite | ...
  |__abbr | ...
  |__acronym | ...
  |__q | ...
  |__tt | ...
  |__i | ...
  |__b | ...
  |__big | ...
  |__small | ...
  |__sub | ...
  |__sup | ...
  |__bdo | ...
  |__a | ...
  |__img | ...
  |__map | ...
  |__object | ...
  |__input | ...
  |__select | ...
  |__textarea | ...
  |__label | ...
  |__button | ...
  |__ruby | ...
  |__ins | ...
  |__del | ...
  |__script | ...
  |__noscript)* ...

15.1.1.3 STRONG

|__strong
  |_(#PCDATA |
  |__br | ...
  |__span | ...
  |__em | ...
  |__strong | ...
  |__dfn | ...
  |__code | ...
  |__samp | ...
  |__kbd | ...
  |__var | ...
  |__cite | ...
  |__abbr | ...
  |__acronym | ...
  |__q | ...
  |__tt | ...
  |__i | ...
  |__b | ...
  |__big | ...
  |__small | ...
  |__sub | ...
  |__sup | ...
  |__bdo | ...
  |__a | ...
  |__img | ...
  |__map | ...
  |__object | ...
  |__input | ...
  |__select | ...
  |__textarea | ...
  |__label | ...
  |__button | ...
  |__ruby | ...
  |__ins | ...
  |__del | ...
  |__script | ...
  |__noscript)* ...

15.1.1.4 CODE

|__code
  |_(#PCDATA |
  |__br | ...
  |__span | ...
  |__em | ...
  |__strong | ...
  |__dfn | ...
  |__code | ...
  |__samp | ...
  |__kbd | ...
  |__var | ...
  |__cite | ...
  |__abbr | ...
  |__acronym | ...
  |__q | ...
  |__tt | ...
  |__i | ...
  |__b | ...
  |__big | ...
  |__small | ...
  |__sub | ...
  |__sup | ...
  |__bdo | ...
  |__a | ...
  |__img | ...
  |__map | ...
  |__object | ...
  |__input | ...
  |__select | ...
  |__textarea | ...
  |__label | ...
  |__button | ...
  |__ruby | ...
  |__ins | ...
  |__del | ...
  |__script | ...
  |__noscript)* ...

15.1.1.5 SAMP

|__*samp*
  |_*(#DATA*
  |__*br* | *...*
  |__*span* | *...*
  |__*em* | *...*
  |__*strong* | *...*
  |__*dfn* | *...*
  |__*code* | *...*
  |__*samp* | *...*
  |__*kbd* | *...*
  |__*var* | *...*
  |__*cite* | *...*
  |__*abbr* | *...*
  |__*acronym* | *...*
  |__*q* | *...*
  |__*tt* | *...*
  |__*i* | *...*
  |__*b* | *...*
  |__*big* | *...*
  |__*small* | *...*
  |__*sub* | *...*
  |__*sup* | *...*
  |__*bdo* | *...*
  |__*a* | *...*
  |__*img* | *...*
  |__*map* | *...*
  |__*object* | *...*
  |__*input* | *...*
  |__*select* | *...*
  |__*textarea* | *...*
  |__*label* | *...*
  |__*button* | *...*
  |__*ruby* | *...*
  |__*ins* | *...*
  |__*del* | *...*
  |__*script* | *...*
  |__*noscript)** *...*

15.1.1.6 CITE

|__cite
  |_(#PCDATA
  |__br | ...
  |__span | ...
  |__em | ...
  |__strong | ...
  |__dfn | ...
  |__code | ...
  |__samp | ...
  |__kbd | ...
  |__var | ...
  |__cite | ...
  |__abbr | ...
  |__acronym | ...
  |__q | ...
  |__tt | ...
  |__i | ...
  |__b | ...
  |__big | ...
  |__small | ...
  |__sub | ...
  |__sup | ...
  |__bdo | ...
  |__a | ...
  |__img | ...
  |__map | ...
  |__object | ...
  |__input | ...
  |__select | ...
  |__textarea | ...
  |__label | ...
  |__button | ...
  |__ruby | ...
  |__ins | ...
  |__del | ...
  |__script | ...
  |__noscript)* ...

15.1.1.7 ABBR

|__abbr
  |_(#PCDATA |
  |__br | ...
  |__span | ...
  |__em | ...
  |__strong | ...
  |__dfn | ...
  |__code | ...
  |__samp | ...
  |__kbd | ...
  |__var | ...
  |__cite | ...
  |__abbr | ...
  |__acronym | ...
  |__q | ...
  |__tt | ...
  |__i | ...
  |__b | ...
  |__big | ...
  |__small | ...
  |__sub | ...
  |__sup | ...
  |__bdo | ...
  |__a | ...
  |__img | ...
  |__map | ...
  |__object | ...
  |__input | ...
  |__select | ...
  |__textarea | ...
  |__label | ...
  |__button | ...
  |__ruby | ...
  |__ins | ...
  |__del | ...
  |__script | ...
  |__noscript)* ...

15.1.1.8 ACRONIM

|__acronym
  |_(#PCDATA |
  |__br | ...
  |__span | ...
  |__em | ...
  |__strong | ...
  |__dfn | ...
  |__code | ...
  |__samp | ...
  |__kbd | ...
  |__var | ...
  |__cite | ...
  |__abbr | ...
  |__acronym | ...
  |__q | ...
  |__tt | ...
  |__i | ...
  |__b | ...
  |__big | ...
  |__small | ...
  |__sub | ...
  |__sup | ...
  |__bdo | ...
  |__a | ...
  |__img | ...
  |__map | ...
  |__object | ...
  |__input | ...
  |__select | ...
  |__textarea | ...
  |__label | ...
  |__button | ...
  |__ruby | ...
  |__ins | ...
  |__del | ...
  |__script | ...
  |__noscript)* ...

15.1.1.9 Q

|__q
  |_(#PCDATA |
  |__br | ...
  |__span | ...
  |__em | ...
  |__strong | ...
  |__dfn | ...
  |__code | ...
  |__samp | ...
  |__kbd | ...
  |__var | ...
  |__cite | ...
  |__abbr | ...
  |__acronym | ...
  |__q | ...
  |__tt | ...
  |__i | ...
  |__b | ...
  |__big | ...
  |__small | ...
  |__sub | ...
  |__sup | ...
  |__bdo | ...
  |__a | ...
  |__img | ...
  |__map | ...
  |__object | ...
  |__input | ...
  |__select | ...
  |__textarea | ...
  |__label | ...
  |__button | ...
  |__ruby | ...
  |__ins | ...
  |__del | ...
  |__script | ...
  |__noscript)* ...

15.1.1.10 TT

|__tt
  |_(#PCDATA |
  |__br | ...
  |__span | ...
  |__em | ...
  |__strong | ...
  |__dfn | ...
  |__code | ...
  |__samp | ...
  |__kbd | ...
  |__var | ...
  |__cite | ...
  |__abbr | ...
  |__acronym | ...
  |__q | ...
  |__tt | ...
  |__i | ...
  |__b | ...
  |__big | ...
  |__small | ...
  |__sub | ...
  |__sup | ...
  |__bdo | ...
  |__a | ...
  |__img | ...
  |__map | ...
  |__object | ...
  |__input | ...
  |__select | ...
  |__textarea | ...
  |__label | ...
  |__button | ...
  |__ruby | ...
  |__ins | ...
  |__del | ...
  |__script | ...
  |__noscript)* ...

15.1.1.11 SUB

|__sub
  |_(#PCDATA |
  |__br | ...
  |__span | ...
  |__em | ...
  |__strong | ...
  |__dfn | ...
  |__code | ...
  |__samp | ...
  |__kbd | ...
  |__var | ...
  |__cite | ...
  |__abbr | ...
  |__acronym | ...
  |__q | ...
  |__tt | ...
  |__i | ...
  |__b | ...
  |__big | ...
  |__small | ...
  |__sub | ...
  |__sup | ...
  |__bdo | ...
  |__a | ...
  |__img | ...
  |__map | ...
  |__object | ...
  |__input | ...
  |__select | ...
  |__textarea | ...
  |__label | ...
  |__button | ...
  |__ruby | ...
  |__ins | ...
  |__del | ...
  |__script | ...
  |__noscript)* ...

15.1.1.12 SUP

|__sup |
  |_(#PCDATA |
  |__br | ...
  |__span | ...
  |__em | ...
  |__strong | ...
  |__dfn | ...
  |__code | ...
  |__samp | ...
  |__kbd | ...
  |__var | ...
  |__cite | ...
  |__abbr | ...
  |__acronym | ...
  |__q | ...
  |__tt | ...
  |__i | ...
  |__b | ...
  |__big | ...
  |__small | ...
  |__sub | ...
  |__sup | ...
  |__bdo | ...
  |__a | ...
  |__img | ...
  |__map | ...
  |__object | ...
  |__input | ...
  |__select | ...
  |__textarea | ...
  |__label | ...
  |__button | ...
  |__ruby | ...
  |__ins | ...
  |__del | ...
  |__script | ...
  |__noscript)* ...

15.1.1.13 A

|__a |
  |_(#PCDATA |
  |__br | ...
  |__span | ...
  |__em | ...
  |__strong | ...
  |__dfn | ...
  |__code | ...
  |__samp | ...
  |__kbd | ...
  |__var | ...
  |__cite | ...
  |__abbr | ...
  |__acronym | ...
  |__q | ...
  |__tt | ...
  |__i | ...
  |__b | ...
  |__big | ...
  |__small | ...
  |__sub | ...
  |__sup | ...
  |__bdo | ...
  |__img | ...
  |__map | ...
  |__object | ...
  |__input | ...
  |__select | ...
  |__textarea | ...
  |__label | ...
  |__button | ...
  |__ruby | ...
  |__ins | ...
  |__del | ...
  |__script | ...
  |__noscript)* ..

15.1.1.14 MAP

|__map |
  |_((h1 | ...
  |___h2 | ...
  |___h3 | ...
  |___h4 | ...
  |___h5 | ...
  |___h6 | ...
  |___ul | ...
  |___ol | ...
  |___dl | ...
  |___p | ...
  |___div | ...
  |___pre | ...
  |___blockquote | ...
  |___address | ...
  |___hr | ...
  |___table | ...
  |___form | ...
  |___fieldset | ...
  |___ins | ...
  |___del | ...
  |___script | ...
  |___noscript) | ...
  |__area)+
    |_EMPTY

15.1.1.15 SELECT

|__select |
  |_(optgroup |
  | |_(option)+ ...
  |
  |__option)+
    |_(#PCDATA)

15.1.1.16 TEXTAREA

|__textarea |
  |_(#PCDATA)

15.1.1.17 LABEL

|__label |
  |_(#PCDATA |
  |__input | ...
  |__select | ...
  |__textarea | ...
  |__button | ...
  |__br | ...
  |__span | ...
  |__em | ...
  |__strong | ...
  |__dfn | ...
  |__code | ...
  |__samp | ...
  |__kbd | ...
  |__var | ...
  |__cite | ...
  |__abbr | ...
  |__acronym | ...
  |__q | ...
  |__bdo | ...
  |__tt | ...
  |__i | ...
  |__b | ...
  |__big | ...
  |__small | ...
  |__sub | ...
  |__sup | ...
  |__a | ...
  |__img | ...
  |__map | ...
  |__object | ...
  |__ins | ...
  |__del | ...
  |__script | ...
  |__noscript)* ...

15.1.1.18 BUTTON

|__button |
  |_(#PCDATA |
  |__h1 | ...
  |__h2 | ...
  |__h3 | ...
  |__h4 | ...
  |__h5 | ...
  |__h6 | ...
  |__ul | ...
  |__ol | ...
  |__dl | ...
  |__p | ...
  |__div | ...
  |__pre | ...
  |__blockquote | ...
  |__address | ...
  |__hr | ...
  |__table | ...
  |__ins | ...
  |__del | ...
  |__script | ...
  |__noscript | ...
  |__br | ...
  |__span | ...
  |__em | ...
  |__strong | ...
  |__dfn | ...
  |__code | ...
  |__samp | ...
  |__kbd | ...
  |__var | ...
  |__cite | ...
  |__abbr | ...
  |__acronym | ...
  |__q | ...
  |__tt | ...
  |__i | ...
  |__b | ...
  |__big | ...
  |__small | ...
  |__sub | ...
  |__sup | ...
  |__bdo | ...
  |__img | ...
  |__map | ...
  |__object)* ...

15.1.1.19 H1, H2, H3, H4, H5, H6

|_(hn |
  |_(#PCDATA |
  |__br | ...
  |__span | ...
  |__em | ...
  |__strong | ...
  |__dfn | ...
  |__code | ...
  |__samp | ...
  |__kbd | ...
  |__var | ...
  |__cite | ...
  |__abbr | ...
  |__acronym | ...
  |__q | ...
  |__tt | ...
  |__i | ...
  |__b | ...
  |__big | ...
  |__small | ...
  |__sub | ...
  |__sup | ...
  |__bdo | ...
  |__a | ...
  |__img | ...
  |__map | ...
  |__object | ...
  |__input | ...
  |__select | ...
  |__textarea | ...
  |__label | ...
  |__button | ...
  |__ruby | ...
  |__ins | ...
  |__del | ...
  |__script | ...
  |__noscript)* ...

15.1.1.20 UL

|__ul |
  |_(li)+
    |_(#PCDATA |
    |__h1 | ...
    |__h2 | ...
    |__h3 | ...
    |__h4 | ...
    |__h5 | ...
    |__h6 | ...
    |__ul | ...
    |__ol | ...
    |__dl | ...
    |__p | ...
    |__div | ...
    |__pre | ...
    |__blockquote | ...
    |__address | ...
    |__hr | ...
    |__table | ...
    |__form | ...
    |__fieldset | ...
    |__br | ...
    |__span | ...
    |__em | ...
    |__strong | ...
    |__dfn | ...
    |__code | ...
    |__samp | ...
    |__kbd | ...
    |__var | ...
    |__cite | ...
    |__abbr | ...
    |__acronym | ...
    |__q | ...
    |__tt | ...
    |__i | ...
    |__b | ...
    |__big | ...
    |__small | ...
    |__sub | ...
    |__sup | ...
    |__bdo | ...
    |__a | ...
    |__img | ...
    |__map | ...
    |__object | ...
    |__input | ...
    |__select | ...

|__textarea | ...
|__label | ...
|__button | ...
|__ruby | ...
|__ins | ...
|__del | ...
|__script | ...
|__noscript)* ...

OL

|__ol |
|_(li)+ ...

DL

|__dl |
|_(dt |....
|
|__dd)+
|_(#PCDATA |....

15.1.1.21 DT

|_(dt |
|_(#PCDATA |
|__br | ...
|__span | ...
|__em | ...
|__strong | ...
|__dfn | ...
|__code | ...
|__samp | ...
|__kbd | ...
|__var | ...
|__cite | ...
|__abbr | ...
|__acronym | ...
|__q | ...
|__tt | ...
|__i | ...
|__b | ...
|__big | ...
|__small | ...
|__sub | ...
|__sup | ...

|__bdo | ...
|__a | ...
|__img | ...
|__map | ...
|__object | ...
|__input | ...
|__select | ...
|__textarea | ...
|__label | ...
|__button | ...
|__ruby | ...
|__ins | ...
|__del | ...
|__script | ...
|__noscript)* ...

15.1.1.22 DD

|__dd)+
|_(#PCDATA |
|__h1 | ...
|__h2 | ...
|__h3 | ...
|__h4 | ...
|__h5 | ...
|__h6 | ...
|__ul | ...
|__ol | ...
|__dl | ...
|__p | ...
|__div | ...
|__pre | ...
|__blockquote | ...
|__address | ...
|__hr | ...
|__table | ...
|__form | ...
|__fieldset | ...
|__br | ...
|__span | ...
|__em | ...
|__strong | ...
|__dfn | ...
|__code | ...
|__samp | ...
|__kbd | ...
|__var | ...
|__cite | ...
|__abbr | ...

|__acronym | ...
|__q | ...
|__tt | ...
|__i | ...
|__b | ...
|__big | ...
|__small | ...
|__sub | ...
|__sup | ...
|__bdo | ...
|__a | ...
|__img | ...
|__map | ...
|__object | ...
|__input | ...
|__select | ...
|__textarea | ...
|__label | ...
|__button | ...
|__ruby | ...
|__ins | ...
|__del | ...
|__script | ...
|__noscript)* ...

15.1.1.23 P

|__p |
  |_(#PCDATA |
  |__br | ...
  |__span | ...
  |__em | ...
  |__strong | ...
  |__dfn | ...
  |__code | ...
  |__samp | ...
  |__kbd | ...
  |__var | ...
  |__cite | ...
  |__abbr | ...
  |__acronym | ...
  |__q | ...
  |__tt | ...
  |__i | ...
  |__b | ...
  |__big | ...
  |__small | ...
  |__sub | ...
  |__sup | ...
  |__bdo | ...
  |__a | ...
  |__img | ...
  |__map | ...
  |__object | ...
  |__input | ...
  |__select | ...
  |__textarea | ...
  |__label | ...
  |__button | ...
  |__ruby | ...
  |__ins | ...
  |__del | ...
  |__script | ...
  |__noscript)* ...

15.1.1.24 DIV

|__div |
  |_(#PCDATA |
  |__h1 | ...
  |__h2 | ...
  |__h3 | ...
  |__h4 | ...

|__h5 | ...
|__h6 | ...
|__ul | ...
|__ol | ...
|__dl | ...
|__p | ...
|__div | ...
|__pre | ...
|__blockquote | ...
|__address | ...
|__hr | ...
|__table | ...
|__form | ...
|__fieldset | ...
|__br | ...
|__span | ...
|__em | ...
|__strong | ...
|__dfn | ...
|__code | ...
|__samp | ...
|__kbd | ...
|__var | ...
|__cite | ...
|__abbr | ...
|__acronym | ...
|__q | ...
|__tt | ...
|__i | ...
|__b | ...
|__big | ...
|__small | ...
|__sub | ...
|__sup | ...
|__bdo | ...
|__a | ...
|__img | ...
|__map | ...
|__object | ...
|__input | ...
|__select | ...
|__textarea | ...
|__label | ...
|__button | ...
|__ruby | ...
|__ins | ...
|__del | ...
|__script | ...
|__noscript)* ...

15.1.1.25 PRE

|__pre |
  |_(#PCDATA |
  |__br | ...
  |__span | ...
  |__em | ...
  |__strong | ...
  |__dfn | ...
  |__code | ...
  |__samp | ...
  |__kbd | ...
  |__var | ...
  |__cite | ...
  |__abbr | ...
  |__acronym | ...
  |__q | ...
  |__tt | ...
  |__i | ...
  |__b | ...
  |__bdo | ...
  |__a | ...
  |__script | ...
  |__map)* ...

15.1.1.26 BLOCKQUOTE

|__blockquote |
  |_(h1 | ...
  |__h2 | ...
  |__h3 | ...
  |__h4 | ...
  |__h5 | ...
  |__h6 | ...
  |__ul | ...
  |__ol | ...
  |__dl | ...
  |__p | ...
  |__div | ...
  |__pre | ...
  |__blockquote | ...
  |__address | ...
  |__hr | ...
  |__table | ...
  |__form | ...
  |__fieldset | ...
  |__ins | ...
  |__del | ...
  |__script | ...

|__noscript)+ ...

15.1.1.27 ADDRESS

|__address |
|_(#PCDATA |
|__br | ...
|__span | ...
|__em | ...
|__strong | ...
|__dfn | ...
|__code | ...
|__samp | ...
|__kbd | ...
|__var | ...
|__cite | ...
|__abbr | ...
|__acronym | ...
|__q | ...
|__tt | ...
|__i | ...
|__b | ...
|__big | ...
|__small | ...
|__sub | ...
|__sup | ...
|__bdo | ...
|__a | ...
|__img | ...
|__map | ...
|__object | ...
|__input | ...
|__select | ...
|__textarea | ...
|__label | ...
|__button | ...
|__ruby | ...
|__ins | ...
|__del | ...
|__script | ...
|__noscript)* ...

15.1.1.28 TABLE

|__table |
|_(caption?, ...
|
|__(col* |
| |_EMPTY

```
   |
   |___colgroup*),
   |    |_(col)* ...
   |
   |__((thead?,
   |     |_(tr)+ ...
   |
   |____tfoot?,
   |     |_(tr)+ ...
   |
   |____tbody+) |
   |     |_(tr)+ ...
   |
   |___(tr+)))
         |_(th |....
         |
         |__td)+
```

15.1.1.29 CAPTION

```
|_(caption?,
   |_(#PCDATA |
   |__br | ...
   |__span | ...
   |__em | ...
   |__strong | ...
   |__dfn | ...
   |__code | ...
   |__samp | ...
   |__kbd | ...
   |__var | ...
   |__cite | ...
   |__abbr | ...
   |__acronym | ...
   |__q | ...
   |__tt | ...
   |__i | ...
   |__b | ...
   |__big | ...
   |__small | ...
   |__sub | ...
   |__sup | ...
   |__bdo | ...
   |__a | ...
   |__img | ...
   |__map | ...
   |__object | ...
   |__input | ...
   |__select | ...
   |__textarea | ...
```

|__label | ...
|__button | ...
|__ruby | ...
|__ins | ...
|__del | ...
|__script | ...
|__noscript)* ...

15.1.1.30 TH

|_(th |
|_(#PCDATA |
|__h1 | ...
|__h2 | ...
|__h3 | ...
|__h4 | ...
|__h5 | ...
|__h6 | ...
|__ul | ...
|__ol | ...
|__dl | ...
|__p | ...
|__div | ...
|__pre | ...
|__blockquote | ...
|__address | ...
|__hr | ...
|__table | ...
|__form | ...
|__fieldset | ...
|__br | ...
|__span | ...
|__em | ...
|__strong | ...
|__dfn | ...
|__code | ...
|__samp | ...
|__kbd | ...
|__var | ...
|__cite | ...
|__abbr | ...
|__acronym | ...
|__q | ...
|__tt | ...
|__i | ...
|__b | ...
|__big | ...
|__small | ...

|__sub | ...
|__sup | ...
|__bdo | ...
|__a | ...
|__img | ...
|__map | ...
|__object | ...
|__input | ...
|__select | ...
|__textarea | ...
|__label | ...
|__button | ...
|__ruby | ...
|__ins | ...
|__del | ...
|__script | ...
|__noscript)* ...

15.1.1.31 TD

|__td)+
|_(#PCDATA |
|__h1 | ...
|__h2 | ...
|__h3 | ...
|__h4 | ...
|__h5 | ...
|__h6 | ...
|__ul | ...
|__ol | ...
|__dl | ...
|__p | ...
|__div | ...
|__pre | ...
|__blockquote | ...
|__address | ...
|__hr | ...
|__table | ...
|__form | ...
|__fieldset | ...
|__br | ...
|__span | ...
|__em | ...
|__strong | ...
|__dfn | ...
|__code | ...
|__samp | ...
|__kbd | ...
|__var | ...
|__cite | ...

|__abbr | ...
|__acronym | ...
|__q | ...
|__tt | ...
|__i | ...
|__b | ...
|__big | ...
|__small | ...
|__sub | ...
|__sup | ...
|__bdo | ...
|__a | ...
|__img | ...
|__map | ...
|__object | ...
|__input | ...
|__select | ...
|__textarea | ...
|__label | ...
|__button | ...
|__ruby | ...
|__ins | ...
|__del | ...
|__script | ...
|__noscript)* ...

15.1.1.32 FORM

|__form |
|_(h1 | ...
|__h2 | ...
|__h3 | ...
|__h4 | ...
|__h5 | ...
|__h6 | ...
|__ul | ...
|__ol | ...
|__dl | ...
|__p | ...
|__div | ...
|__pre | ...
|__blockquote | ...
|__address | ...
|__hr | ...
|__table | ...
|__ins | ...
|__del | ...
|__script | ...
|__noscript | ...

|__fieldset)+ ...

15.1.1.33 FIELDSET

|__fieldset |
|_(#PCDATA |
|__legend | ....
|__h1 | ...
|__h2 | ...
|__h3 | ...
|__h4 | ...
|__h5 | ...
|__h6 | ...
|__ul | ...
|__ol | ...
|__dl | ...
|__p | ...
|__div | ...
|__pre | ...
|__blockquote | ...
|__address | ...
|__hr | ...
|__table | ...
|__form | ...
|__fieldset | ...
|__br | ...
|__span | ...
|__em | ...
|__strong | ...
|__dfn | ...
|__code | ...
|__samp | ...
|__kbd | ...
|__var | ...
|__cite | ...
|__abbr | ...
|__acronym | ...
|__q | ...
|__tt | ...
|__i | ...
|__b | ...
|__big | ...
|__small | ...
|__sub | ...
|__sup | ...
|__bdo | ...
|__a | ...
|__img | ...

|__map | ...
|__object | ...
|__input | ...
|__select | ...
|__textarea | ...
|__label | ...
|__button | ...
|__ruby | ...
|__ins | ...
|__del | ...
|__script | ...
|__noscript)* ...

15.1.1.34 NOSCRIPT

|__noscript)+
|_(h1 | ...
|__h2 | ...
|__h3 | ...
|__h4 | ...
|__h5 | ...
|__h6 | ...
|__ul | ...
|__ol | ...
|__dl | ...
|__p | ...
|__div | ...
|__pre | ...
|__blockquote | ...
|__address | ...
|__hr | ...
|__table | ...
|__form | ...
|__fieldset | ...
|__ins | ...
|__del | ...
|__script | ...
|__noscript)+ ...

16 Appendice B: CARATTERI SPECIALI

16.1 Caratteri Latin-1

<!ENTITY nbsp " ">
<!ENTITY iexcl "¡"> ¡
<!ENTITY cent "¢"> ¢
<!ENTITY pound "£"> £
<!ENTITY curren "¤"> ¤
<!ENTITY yen "¥"> ¥
<!ENTITY brvbar "¦"> ¦
<!ENTITY sect "§"> §
<!ENTITY uml "¨"> ¨
<!ENTITY copy "©"> ©
<!ENTITY ordf "ª"> ª
<!ENTITY laquo "«"> «
<!ENTITY not "¬"> ¬
<!ENTITY reg "®"> ®
<!ENTITY macr "¯"> ¯
<!ENTITY deg "°"> °
<!ENTITY plusmn "±"> ±
<!ENTITY sup2 "²"> ²
<!ENTITY sup3 "³"> ³
<!ENTITY acute "´"> ´
<!ENTITY micro "µ"> µ
<!ENTITY para "¶"> ¶
<!ENTITY middot "·"> ·
<!ENTITY cedil "¸"> ¸
<!ENTITY sup1 "¹"> ¹
<!ENTITY ordm "º"> º
<!ENTITY raquo "»"> »
<!ENTITY frac14 "¼"> ¼
<!ENTITY frac12 "½"> ½
<!ENTITY frac34 "¾"> ¾
<!ENTITY iquest "¿"> ¿
<!ENTITY Agrave "À"> À
<!ENTITY Aacute "Á"> Á
<!ENTITY Acirc "Â"> Â
<!ENTITY Atilde "Ã"> Ã
<!ENTITY Auml "Ä"> Ä
<!ENTITY Aring "Å"> Å
<!ENTITY AElig "Æ"> Æ
<!ENTITY Ccedil "Ç"> Ç
<!ENTITY Egrave "È"> È
<!ENTITY Eacute "É"> É
<!ENTITY Ecirc "Ê"> Ê
<!ENTITY Euml "Ë"> Ë
<!ENTITY Igrave "Ì"> Ì
<!ENTITY Iacute "Í"> Í
<!ENTITY Icirc "Î"> Î

<!ENTITY Iuml "Ï"> Ï
<!ENTITY ETH "Ð"> Ð
<!ENTITY Ntilde "Ñ"> Ñ
<!ENTITY Ograve "Ò"> Ò
<!ENTITY Oacute "Ó"> Ó
<!ENTITY Ocirc "Ô"> Ô
<!ENTITY Otilde "Õ"> Õ
<!ENTITY Ouml "Ö"> Ö
<!ENTITY times "×"> ×
<!ENTITY Oslash "Ø"> Ø
<!ENTITY Ugrave "Ù"> Ù
<!ENTITY Uacute "Ú"> Ú
<!ENTITY Ucirc "Û"> Û
<!ENTITY Uuml "Ü"> Ü
<!ENTITY Yacute "Ý"> Ý
<!ENTITY THORN "Þ"> Þ
<!ENTITY szlig "ß"> ß
<!ENTITY agrave "à"> à
<!ENTITY aacute "á"> á
<!ENTITY acirc "â"> â
<!ENTITY atilde "ã"> ã
<!ENTITY auml "ä"> ä
<!ENTITY aring "å"> å
<!ENTITY aelig "æ"> æ
<!ENTITY ccedil "ç"> ç
<!ENTITY egrave "è"> è
<!ENTITY eacute "é"> é
<!ENTITY ecirc "ê"> ê
<!ENTITY euml "ë"> ë
<!ENTITY igrave "ì"> ì
<!ENTITY iacute "í"> í
<!ENTITY icirc "î"> î
<!ENTITY iuml "ï"> ï
<!ENTITY eth "ð"> ð
<!ENTITY ograve "ò"> ò
<!ENTITY oacute "ó"> ó
<!ENTITY ocirc "ô"> ô
<!ENTITY otilde "õ"> õ
<!ENTITY ouml "ö"> ö
<!ENTITY divide "÷"> ÷
<!ENTITY oslash "ø"> ø
<!ENTITY uacute "ú"> ú
<!ENTITY ucirc "û"> û
<!ENTITY uuml "ü"> ü
<!ENTITY yacute "ý"> ý
<!ENTITY thorn "þ"> þ
<!ENTITY yuml "ÿ"> ÿ

16.2 Caratteri Speciali

<!ENTITY quot """>	"
<!ENTITY amp "&">	&
<!ENTITY lt "<">	<
<!ENTITY gt ">">	>
<!ENTITY apos "'">	'
<!ENTITY OElig "Œ">	Œ
<!ENTITY oelig "œ">	œ
<!ENTITY Scaron "Š">	Š
<!ENTITY scaron "š">	š
<!ENTITY Yuml "Ÿ">	Ÿ
<!ENTITY circ "ˆ">	ˆ
<!ENTITY tilde "˜">	˜
<!ENTITY thinsp " ">	□
<!ENTITY ndash "–">	–
<!ENTITY mdash "—">	—
<!ENTITY lsquo "‘">	‘
<!ENTITY rsquo "’">	’
<!ENTITY sbquo "‚">	,
<!ENTITY ldquo "“">	“
<!ENTITY rdquo "”">	”
<!ENTITY bdquo "„">	„
<!ENTITY dagger "†">	†
<!ENTITY Dagger "‡">	‡
<!ENTITY permil "‰">	‰
<!ENTITY lsaquo "‹">	‹
<!ENTITY rsaquo "›">	›
<!ENTITY euro "€">	€

16.3 Symbols

<!ENTITY fnof "ƒ"> ƒ

16.3.1 Greek

<!ENTITY Alpha "Α"> Α
<!ENTITY Beta "Β"> Β
<!ENTITY Gamma "Γ"> Γ
<!ENTITY Delta "Δ"> Δ
<!ENTITY Epsilon "Ε"> Ε
<!ENTITY Zeta "Ζ"> Ζ
<!ENTITY Eta "Η"> Η
<!ENTITY Theta "Θ"> Θ
<!ENTITY Iota "Ι"> Ι
<!ENTITY Kappa "Κ"> Κ
<!ENTITY Lambda "Λ"> Λ
<!ENTITY Mu "Μ"> Μ
<!ENTITY Nu "Ν"> Ν
<!ENTITY Xi "Ξ"> Ξ
<!ENTITY Omicron "Ο"> Ο
<!ENTITY Pi "Π"> Π
<!ENTITY Rho "Ρ"> Ρ
<!ENTITY Sigma "Σ"> Σ
<!ENTITY Tau "Τ"> Τ
<!ENTITY Upsilon "Υ"> Υ
<!ENTITY Phi "Φ"> Φ
<!ENTITY Chi "Χ"> Χ
<!ENTITY Psi "Ψ"> Ψ
<!ENTITY Omega "Ω"> Ω
<!ENTITY alpha "α"> α
<!ENTITY beta "β"> β
<!ENTITY gamma "γ"> γ
<!ENTITY delta "δ"> δ
<!ENTITY epsilon "ε"> ε
<!ENTITY zeta "ζ"> ζ
<!ENTITY eta "η"> η
<!ENTITY theta "θ"> θ
<!ENTITY iota "ι"> ι
<!ENTITY kappa "κ"> κ
<!ENTITY lambda "λ"> λ
<!ENTITY mu "μ"> μ
<!ENTITY nu "ν"> ν

<!ENTITY xi "ξ"> ξ
<!ENTITY omicron "ο"> ο
<!ENTITY pi "π"> π
<!ENTITY rho "ρ"> ρ
<!ENTITY sigmaf "ς"> ς
<!ENTITY sigma "σ"> σ
<!ENTITY tau "τ"> τ
<!ENTITY upsilon "υ"> υ
<!ENTITY phi "φ"> φ
<!ENTITY chi "χ"> χ
<!ENTITY psi "ψ"> ψ
<!ENTITY omega "ω"> ω

16.3.2 General Punctuation

<!ENTITY hellip "…"> …
<!ENTITY prime "′"> ′
<!ENTITY Prime "″"> ″
<!ENTITY oline "‾"> ‾
<!ENTITY frasl "⁄"> ⁄

16.3.3 Letterlike Symbols

<!ENTITY trade "™"> ™

16.3.4 Arrows

<!ENTITY larr "←"> ←
<!ENTITY uarr "↑"> ↑
<!ENTITY rarr "→"> →
<!ENTITY darr "↓"> ↓
<!ENTITY harr "↔"> ↔

16.3.5 Mathematical Operators

<!ENTITY part "∂"> ∂
<!ENTITY prod "∏"> ∏
<!ENTITY sum "∑"> ∑
<!ENTITY minus "−"> −
<!ENTITY radic "√"> √
<!ENTITY infin "∞"> ∞
<!ENTITY cap "∩"> ∩

<!ENTITY int "∫"> ∫
<!ENTITY cong "≅"> □
<!ENTITY asymp "≈"> ≈
<!ENTITY ne "≠"> ≠
<!ENTITY equiv "≡"> ≡
<!ENTITY le "≤"> ≤
<!ENTITY ge "≥"> ≥

16.3.6 Geometric Shapes

<!ENTITY loz "◊"> ◊

16.3.7 Miscellaneous Symbols

<!ENTITY spades "♠"> ♠
<!ENTITY clubs "♣"> ♣
<!ENTITY hearts "♥"> ♥
<!ENTITY diams "♦"> ♦

17 appendice C: proprietà CSS

'border-top-style' 'border-right-style' 'border-bottom-style' 'border-left-style'	<border-style> | inherit	none		no		visual
'border-top-width' 'border-right-width' 'border-bottom-width' 'border-left-width'	<border-width> | inherit	medium		no		visual
'border-width'	<border-width>{1,4} | inherit	see individual properties		no		visual
'bottom'	<length> | <percentage> | auto | inherit	auto	positioned elements	no	refer to height of containing block	visual
'caption-side'	top | bottom | left | right | inherit	top	'table-caption' elements	yes		visual
'clear'	none | left | right | both | inherit	none	block-level elements	no		visual
'clip'	<shape> | auto | inherit	auto	block-level and replaced elements	no		visual
'color'	<color> | inherit	depends on user		yes		visual

		agent				
'content'	[<string> | <uri> | <counter> | attr(X) | open-quote | close-quote | no-open-quote | no-close-quote]+ | inherit	empty string	:before and :after pseudo-elements	no		all
'counter-increment'	[<identifier> <integer>?]+ | none | inherit	none		no		all
'counter-reset'	[<identifier> <integer>?]+ | none | inherit	none		no		all
'cue'	['cue-before' || 'cue-after'] | inherit	XX		no		aural
'cue-after'	<uri> | none | inherit	none		no		aural
'cue-before'	<uri> | none | inherit	none		no		aural
'cursor'	[[<uri> ,]* [auto | crosshair | default | pointer | move | e-resize | ne-resize | nw-resize | n-resize | se-resize | sw-resize | s-resize | w-resize| text | wait | help]] | inherit	auto		yes		visual, interactive
'direction'	ltr | rtl | inherit	ltr	all elements, but see prose	yes		visual
'display'	inline | block | list-item | run-in | compact | marker | table | inline-table | table-row-group | table-header-	inline		no		all

	group | table-footer-group | table-row | table-column-group | table-column | table-cell | table-caption | none | inherit					
'elevation'	<angle> | below | level | above | higher | lower | inherit	level		yes		aural
'empty-cells'	show | hide | inherit	show	'table-cell' elements	yes		visual
'float'	left | right | none | inherit	none	all but positioned elements and generated content	no		visual
'font'	[['font-style' || 'font-variant' || 'font-weight']? 'font-size' [/ 'line-height']? 'font-family'] | caption | icon | menu | message-box | small-caption | status-bar | inherit	see individual properties		yes	allowed on 'font-size' and 'line-height'	visual
'font-family'	[[<family-name> | <generic-family>],]* [<family-name> | <generic-family>] | inherit	depends on user agent		yes		visual
'font-size'	<absolute-size> | <relative-size> | <length> |	medium		yes, the computed	refer to parent eleme	visual

	<percentage> | inherit			value is inherited	nt's font size	
'font-size-adjust'	<number> | none | inherit	none		yes		visual
'font-stretch'	normal | wider | narrower | ultra-condensed | extra-condensed | condensed | semi-condensed | semi-expanded | expanded | extra-expanded | ultra-expanded | inherit	normal		yes		visual
'font-style'	normal | italic | oblique | inherit	normal		yes		visual
'font-variant'	normal | small-caps | inherit	normal		yes		visual
'font-weight'	normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | inherit	normal		yes		visual
'height'	<length> | <percentage> | auto | inherit	auto	all elements but non-replaced inline elements, table columns, and column groups	no	see prose	visual

'left'	<length> | <percentage> | auto | inherit	auto	positioned elements	no	refer to width of containing block	visual
'letter-spacing'	normal | <length> | inherit	normal		yes		visual
'line-height'	normal | <number> | <length> | <percentage> | inherit	normal		yes	refer to the font size of the element itself	visual
'list-style'	['list-style-type' || 'list-style-position' || 'list-style-image'] | inherit	XX	elements with 'display: list-item'	yes		visual
'list-style-image'	<uri> | none | inherit	none	elements with 'display: list-item'	yes		visual
'list-style-position'	inside | outside | inherit	outside	elements with 'display: list-item'	yes		visual
'list-style-type'	disc | circle | square | decimal | decimal-leading-zero | lower-roman | upper-roman | lower-greek | lower-alpha | lower-latin | upper-alpha | upper-latin | hebrew | armenian | georgian | cjk-ideographic | hiragana | katakana |	disc	elements with 'display: list-item'	yes		visual

	hiragana-iroha | katakana-iroha | none | inherit					
'margin'	<margin-width>{1,4} | inherit	XX		no	refer to width of containing block	visual
'margin-top' 'margin-right' 'margin-bottom' 'margin-left'	<margin-width> | inherit	0		no	refer to width of containing block	visual
'marker-offset'	<length> | auto | inherit	auto	elements with 'display: marker'	no		visual
'marks'	[crop || cross] | none | inherit	none	page context	N/A		visual, paged
'max-height'	<length> | <percentage> | none | inherit	none	all elements except non-replaced inline elements and table elements	no	refer to height of containing block	visual
'max-width'	<length> | <percentage> | none | inherit	none	all elements except non-replaced	no	refer to width of containing block	visual

			inline elements and table elements			
'min-height'	<length> | <percentage> | inherit	0	all elements except non-replaced inline elements and table elements	no	refer to height of containing block	visual
'min-width'	<length> | <percentage> | inherit	UA dependent	all elements except non-replaced inline elements and table elements	no	refer to width of containing block	visual
'orphans'	<integer> | inherit	2	block-level elements	yes		visual, paged
'outline'	['outline-color' || 'outline-style' || 'outline-width'] | inherit	see individual properties		no		visual, interactive
'outline-color'	<color> | invert | inherit	invert		no		visual, int

						eractive
'outline-style'	<border-style> | inherit	none		no		visual, interactive
'outline-width'	<border-width> | inherit	medium		no		visual, interactive
'overflow'	visible | hidden | scroll | auto | inherit	visible	block-level and replaced elements	no		visual
'padding'	<padding-width>{1,4} | inherit	XX		no	refer to width of containing block	visual
'padding-top' 'padding-right' 'padding-bottom' 'padding-left'	<padding-width> | inherit	0		no	refer to width of containing block	visual
'page'	<identifier> | auto	auto	block-level elements	yes		visual, paged
'page-break-after'	auto | always | avoid | left | right | inherit	auto	block-level elements	no		visual, paged
'page-break-before'	auto | always | avoid | left | right	auto	block-level	no		visual,

	| inherit		elements			paged
'page-break-inside'	avoid | auto | inherit	auto	block-level elements	yes		visual, paged
'pause'	[[<time> | <percentage>]{1,2}] | inherit	depends on user agent		no	see descriptions of 'pause-before' and 'pause-after'	aural
'pause-after'	<time> | <percentage> | inherit	depends on user agent		no	see prose	aural
'pause-before'	<time> | <percentage> | inherit	depends on user agent		no	see prose	aural
'pitch'	<frequency> | x-low | low | medium | high | x-high | inherit	medium		yes		aural
'pitch-range'	<number> | inherit	50		yes		aural
'play-during'	<uri> mix? repeat? | auto | none | inherit	auto		no		aural
'position'	static | relative | absolute | fixed | inherit	static	all elements, but not to generated content	no		visual
'quotes'	[<string> <string>]+ | none | inherit	depends on user agent		yes		visual
'richness'	<number> | inherit	50		yes		aural
'right'	<length> |	auto	positione	no	refer to	visual

	<percentage> | auto | inherit		d elements		width of containing block	
'size'	<length>{1,2} | auto | portrait | landscape | inherit	auto	the page context	N/A		visual, paged
'speak'	normal | none | spell-out | inherit	normal		yes		aural
'speak-header'	once | always | inherit	once	elements that have table header information	yes		aural
'speak-numeral'	digits | continuous | inherit	continuous		yes		aural
'speak-punctuation'	code | none | inherit	none		yes		aural
'speech-rate'	<number> | x-slow | slow | medium | fast | x-fast | faster | slower | inherit	medium		yes		aural
'stress'	<number> | inherit	50		yes		aural
'table-layout'	auto | fixed | inherit	auto	'table' and 'inline-table' elements	no		visual
'text-align'	left | right | center | justify | <string> | inherit	depends on user agent and writing direction	block-level elements	yes		visual
'text-decoration'	none | [underline || overline || line-through || blink]	none		no (see prose)		visual

	| inherit					
'text-indent'	<length> | <percentage> | inherit	0	block-level elements	yes	refer to width of containing block	visual
'text-shadow'	none | [<color> || <length> <length> <length>? ,]* [<color> || <length> <length> <length>?] | inherit	none		no (see prose)		visual
'text-transform'	capitalize | uppercase | lowercase | none | inherit	none		yes		visual
'top'	<length> | <percentage> | auto | inherit	auto	positioned elements	no	refer to height of containing block	visual
'unicode-bidi'	normal | embed | bidi-override | inherit	normal	all elements, but see prose	no		visual
'vertical-align'	baseline | sub | super | top | text-top | middle | bottom | text-bottom | <percentage> | <length> | inherit	baseline	inline-level and 'table-cell' elements	no	refer to the 'line-height' of the element itself	visual
'visibility'	visible | hidden | collapse | inherit	inherit		no		visual
'voice-family'	[[<specific-voice> | <generic-voice>],]* [<specific-voice> | <generic-voice>	depends on user agent		yes		aural

	] | inherit					
'volume'	<number> | <percentage> | silent | x-soft | soft | medium | loud | x-loud | inherit	medium		yes	refer to inherited value	aural
'white-space'	normal | pre | nowrap | inherit	normal	block-level elements	yes		visual
'widows'	<integer> | inherit	2	block-level elements	yes		visual, paged
'width'	<length> | <percentage> | auto | inherit	auto	all elements but non-replaced inline elements, table rows, and row groups	no	refer to width of containing block	visual
'word-spacing'	normal | <length> | inherit	normal		yes		visual
'z-index'	auto | <integer> | inherit	auto	positioned elements	no		visual

18 Appendice D: COLORI

18 Appendice D: COLORI

18.1 Colori

Named	Numeric	Color name	Hex rgb	Decimal
		aliceblue	#f0f8ff	240,248,255
		antiquewhite	#faebd7	250,235,215
		aqua	#00ffff	0,255,255
		aquamarine	#7fffd4	127,255,212
		azure	#f0ffff	240,255,255
		beige	#f5f5dc	245,245,220
		bisque	#ffe4c4	255,228,196
		black	#000000	0,0,0
		blanchedalmond	#ffebcd	255,235,205
		blue	#0000ff	0,0,255
		blueviolet	#8a2be2	138,43,226
		brown	#a52a2a	165,42,42
		burlywood	#deb887	222,184,135
		cadetblue	#5f9ea0	95,158,160
		chartreuse	#7fff00	127,255,0
		chocolate	#d2691e	210,105,30
		coral	#ff7f50	255,127,80
		cornflowerblue	#6495ed	100,149,237
		cornsilk	#fff8dc	255,248,220
		crimson	#dc143c	220,20,60
		cyan	#00ffff	0,255,255
		darkblue	#00008b	0,0,139
		darkcyan	#008b8b	0,139,139
		darkgoldenrod	#b8860b	184,134,11
		darkgray	#a9a9a9	169,169,169
		darkgreen	#006400	0,100,0
		darkgrey	#a9a9a9	169,169,169

darkkhaki	#bdb76b	189,183,107
darkmagenta	#8b008b	139,0,139
darkolivegreen	#556b2f	85,107,47
darkorange	#ff8c00	255,140,0
darkorchid	#9932cc	153,50,204
darkred	#8b0000	139,0,0
darksalmon	#e9967a	233,150,122
darkseagreen	#8fbc8f	143,188,143
darkslateblue	#483d8b	72,61,139
darkslategray	#2f4f4f	47,79,79
darkslategrey	#2f4f4f	47,79,79
darkturquoise	#00ced1	0,206,209
darkviolet	#9400d3	148,0,211
deeppink	#ff1493	255,20,147
deepskyblue	#00bfff	0,191,255
dimgray	#696969	105,105,105
dimgrey	#696969	105,105,105
dodgerblue	#1e90ff	30,144,255
firebrick	#b22222	178,34,34
floralwhite	#fffaf0	255,250,240
forestgreen	#228b22	34,139,34
fuchsia	#ff00ff	255,0,255
gainsboro	#dcdcdc	220,220,220
ghostwhite	#f8f8ff	248,248,255
gold	#ffd700	255,215,0
goldenrod	#daa520	218,165,32
gray	#808080	128,128,128
green	#008000	0,128,0
greenyellow	#adff2f	173,255,47
grey	#808080	128,128,128
honeydew	#f0fff0	240,255,240
hotpink	#ff69b4	255,105,180

indianred	#cd5c5c	205,92,92
indigo	#4b0082	75,0,130
ivory	#fffff0	255,255,240
khaki	#f0e68c	240,230,140
lavender	#e6e6fa	230,230,250
lavenderblush	#fff0f5	255,240,245
lawngreen	#7cfc00	124,252,0
lemonchiffon	#fffacd	255,250,205
lightblue	#add8e6	173,216,230
lightcoral	#f08080	240,128,128
lightcyan	#e0ffff	224,255,255
lightgoldenrodyellow	#fafad2	250,250,210
lightgray	#d3d3d3	211,211,211
lightgreen	#90ee90	144,238,144
lightgrey	#d3d3d3	211,211,211
lightpink	#ffb6c1	255,182,193
lightsalmon	#ffa07a	255,160,122
lightseagreen	#20b2aa	32,178,170
lightskyblue	#87cefa	135,206,250
lightslategray	#778899	119,136,153
lightslategrey	#778899	119,136,153
lightsteelblue	#b0c4de	176,196,222
lightyellow	#ffffe0	255,255,224
lime	#00ff00	0,255,0
limegreen	#32cd32	50,205,50
linen	#faf0e6	250,240,230
magenta	#ff00ff	255,0,255
maroon	#800000	128,0,0
mediumaquamarine	#66cdaa	102,205,170
mediumblue	#0000cd	0,0,205
mediumorchid	#ba55d3	186,85,211
mediumpurple	#9370db	147,112,219

mediumseagreen	#3cb371	60,179,113
mediumslateblue	#7b68ee	123,104,238
mediumspringgreen	#00fa9a	0,250,154
mediumturquoise	#48d1cc	72,209,204
mediumvioletred	#c71585	199,21,133
midnightblue	#191970	25,25,112
mintcream	#f5fffa	245,255,250
mistyrose	#ffe4e1	255,228,225
moccasin	#ffe4b5	255,228,181
navajowhite	#ffdead	255,222,173
navy	#000080	0,0,128
oldlace	#fdf5e6	253,245,230
olive	#808000	128,128,0
olivedrab	#6b8e23	107,142,35
orange	#ffa500	255,165,0
orangered	#ff4500	255,69,0
orchid	#da70d6	218,112,214
palegoldenrod	#eee8aa	238,232,170
palegreen	#98fb98	152,251,152
paleturquoise	#afeeee	175,238,238
palevioletred	#db7093	219,112,147
papayawhip	#ffefd5	255,239,213
peachpuff	#ffdab9	255,218,185
peru	#cd853f	205,133,63
pink	#ffc0cb	255,192,203
plum	#dda0dd	221,160,221
powderblue	#b0e0e6	176,224,230
purple	#800080	128,0,128
red	#ff0000	255,0,0
rosybrown	#bc8f8f	188,143,143
royalblue	#4169e1	65,105,225
saddlebrown	#8b4513	139,69,19

salmon	#fa8072	250,128,114
sandybrown	#f4a460	244,164,96
seagreen	#2e8b57	46,139,87
seashell	#fff5ee	255,245,238
sienna	#a0522d	160,82,45
silver	#c0c0c0	192,192,192
skyblue	#87ceeb	135,206,235
slateblue	#6a5acd	106,90,205
slategray	#708090	112,128,144
slategrey	#708090	112,128,144
snow	#fffafa	255,250,250
springgreen	#00ff7f	0,255,127
steelblue	#4682b4	70,130,180
tan	#d2b48c	210,180,140
teal	#008080	0,128,128
thistle	#d8bfd8	216,191,216
tomato	#ff6347	255,99,71
turquoise	#40e0d0	64,224,208
violet	#ee82ee	238,130,238
wheat	#f5deb3	245,222,179
white	#ffffff	255,255,255
whitesmoke	#f5f5f5	245,245,245
yellow	#ffff00	255,255,0
yellowgreen	#9acd32	154,205,50

Bibliografia & approfondimenti

Capitolo 2 – Il servizio WWW

[AHUCF] AUGMENTING HUMAN INTELLECT: A Conceptual Framework http://www.bootstrap.org/augdocs/friedewald030402/augmentinghumanintellect/ahi62index.html

[APACHE] The Apache HTTP Server Project http://httpd.apache.org/

[Berners-Lee] Home page Tim Berners-Lee http://www.w3.org/People/Berners-Lee/

[DSSSL] ISO/IEC 10179:1996 Document Style Semantics and Specification Language (DSSSL) http://www.jclark.com/dsssl/

[ENGELBART] Internet Pioneers: Doug Engelbart http://www.ibiblio.org/pioneers/engelbart.html

[IANA] Internet Assigned Numbers Authority (IANA) - Home page http://www.iana.org/

[ISOC] Internet Society (Isoc) http://www.isoc.org/

[MOZILLA] Mozilla project - Home page http://www.mozilla.org

[NCSA] The National Center for Supercomputing Applications (NCSA) http://www.ncsa.uiuc.edu/

[NELSON] Internet Pioneers: Ted Nelson http://www.ibiblio.org/pioneers/nelson.html

[NLS] Wikipedia: NLS (computer system) http://en.wikipedia.org/wiki/NLS_(computer_system)

[NLSDEMO] NLS Demo http://sloan.stanford.edu/MouseSite/1968Demo.html

[POSTEL] Wikipedia: Jon Postel http://en.wikipedia.org/wiki/Jon_Postel

[RFC] Request for Comments (RFC) http://www.ietf.org/rfc.html

[RFC2616] Hypertext Transfer Protocol -- HTTP/1.1, giugno 1999 http://www.ietf.org/rfc/rfc2616.txt

[RFC3986] Uniform Resource Identifier (URI): Generic Syntax, gennaio 2005 http://www.ietf.org/rfc/rfc3986.txt

[SGML] Standard Generalized Markup Language (SGML) http://xml.coverpages.org/sgml.html

[URISCHEMES] Uniform Resource Identifier (URI) SCHEMES http://www.iana.org/assignments/uri-schemes

[URI-URL-URN] URIs, URLs, and URNs: Clarifications and Recommendations 1.0, 21 settembre 2001 Report from the joint W3C/IETF URI Planning Interest Group http://www.w3.org/TR/uri-clarification/

[VIOLA] Viola Home Page http://www.viola.org/

[XANADU] PROJECT XANADU - Founded 1960 * The Original Hypertext Project http://www.xanadu.net/

[XML] Extensible Markup Language (XML) http://www.w3.org/XML/

[XSL] The Extensible Stylesheet Language Family (XSL) http://www.w3.org/Style/XSL/

[W3C] World Wide Web consortium - Home page http://www.w3.org

Capitolo 3 – Progettare: parte 1

Barrett E., Levinson D.A., Lisanti S., *The MIT guide to teaching web site design*, The MIT Press, Cambridge, MA & London, UK, 2001

Cantoni L., Di Blas N., *Teorie e pratiche della comunicazione*, Apogeo, Milano, 2002

Goto Kelly, Cotler Emily, *Web ReDesign. Strumenti e metodi per la riprogettazione di un sito Web*, Apogeo, Milano, 2002 (vers.orig. Web ReDesign: workflow that Works. New Riders)

Kanizsa G., *La grammatica del vedere*, Il Mulino, Bologna, 1980

Krug Steve, Don't make me think. Un approccio di buon senso all'usabilità web. HOPS Libri, 2000

Wilde Richard and Judith, *Visual Literacy*, New York: Watson-Guptill Pubblications, 1991

Williams Robin, *The non-designer's design book. Design and typography principles for visual novice*, 2° ed. Peachpit Press, Berkeley, 2004

http://archimedia.polito.it/multimedialita/tipologia.html: Maggipinto A., Classificazione tassonomia e specificità dei siti, 2001

http://www.useit.com/alertbox/20030331.html: Nielsen J., Intranet Portals: A Tool Metaphor for Corporate Information, 2003

Capitolo 5 – HTML: struttura di una pagina

"XHTML Media Types", Masayasu Ishikawa, 1 August 2002. Latest version available at: http://www.w3.org/TR/xhtml-media-types

XHTML 1.0: The Extensible HyperText Markup Language, W3C Recommendation, Steven Pemberton, et al., 26 January 2000. See: http://www.w3.org/TR/2000/REC-xhtml1-20000126

RFC1766:Tags for the Identification of Languages, H. Alvestrand, March 1995. http://www.ietf.org/rfc/rfc1766.txt

RFC1521: MIME (Multipurpose Internet Mail Extensions) Mechanisms for Specifying and Describing the Format of Internet Message Bodies, N. Borenstein, N. Freed, September 1993. http://www.ietf.org/rfc/rfc1521.txt

XHTML - Chelsea Valentine, Chris Minnick della ADDISON-WESLEY (ISBN 88-7192-110-0) (libro che mi hai prestato te ... non so se il riferimento va bene)

Capitolo 7 – Altri elementi HTML

"RFC2396: Uniform Resource Identifiers (URI): Generic Syntax", T. Berners-Lee, R. Fielding, L. Masinter, August 1998. This document updates RFC1738 and RFC1808. Available at: http://www.ietf.org/rfc/rfc2396.txt

"XHTML Media Types", Masayasu Ishikawa, 1 August 2002. Latest version available at: http://www.w3.org/TR/xhtml-media-types

XHTML 1.0: The Extensible HyperText Markup Language, W3C Recommendation, Steven Pemberton, et al., 26 January 2000. See: http://www.w3.org/TR/2000/REC-xhtml1-20000126

RFC1766:Tags for the Identification of Languages, H. Alvestrand, March 1995. http://www.ietf.org/rfc/rfc1766.txt

RFC1521: MIME (Multipurpose Internet Mail Extensions) Mechanisms for Specifying and Describing the Format of Internet Message Bodies, N. Borenstein, N. Freed, September 1993. http://www.ietf.org/rfc/rfc1521.txt

XHTML - Chelsea Valentine, Chris Minnick della ADDISON-WESLEY (ISBN 88-7192-110-0)

Capitolo 10 – Progettare: parte 2

Carroll J.M., *Scenario-Based Design*, John Wiley & Sons, New York, 1995

Nielsen J., *Web usability*. Apogeo, Milano, 2000 (vers.orig. Designing Web Usability. Macmillian Computer Publishing)

Nielsen J., *Usability engineering*, Academic Press, 1993

Norman Donald A., *Le cose che ci fanno intelligenti*, Feltrinelli, Bologna, 1995 (vers.orig. Things that make us smart. Addison Wesley. 1995)

Norman Donald A., *La caffettiera del masochista. Psicopatologia degli oggetti quotidiani*, Giunti, Firenze, 1997 (vers.orig. The psychology of everyday things. Basic Books Inc. New York, 1988)

Norman Donald A., Draper S.W., *User Centered System Design. New perspectives on Human-Computer Interaction*, LEA, Hillsdale & London, 1986

Monk Andrew, *Fundamentals of Human-Computer Interaction*. Academic Press Inc., London, 1985

Pearrow Mark, *Web Site Usability Handbook*, Charles River Media, 2000

Preece Jenny, *Human-Computer Interaction*. Addison Wesley, 1994

Rizzo A., *Gli artefatti e la loro progettazione*, Sistemi Intelligenti. XII n.3, pp 437-452; 2000

Rizzo A., Marti P., Bagnara S., *L'Interazione Uomo-Computer*. In E. Burattini & R. Cordeschi (A cura di) *Manuale di Intelligenza Artificiale*. Roma: Carocci, 2001

Shneiderman Ben, *Designing the user interface. Strategies for effective Human-Computer Interaction*, Addison-Wesley, 1987

www.useit.com: il sito ufficiale di Jakob Nielsen, considerato il padre dell'usabilità dei siti web, in cui è possibile trovare la lista compleata di tutti gli Alertbox ordinate per data di uscita

www.usableweb.com; si trova una vasta collezione di link ai siti web sull'usabilità e viene aggiornato da Keith Instone

www.usablesites.com: è il sito di riferimento di Mark Pearrow, aggiornato da lui e dai suoi studenti

http://www.grc.nasa.gov/WWW/usability/processcss.html: User-Centered Design Process della NASA, da cui poter scaricare la lista di attività da portare a termine in un processo UCD

http://www.usability.serco.com/trump/resources/: sito della Serco Ltd., UK, con risorse utile sulla metodologia UCD

Capitolo 13 – Progettare: parte 3

http://www.webxtutti.it/testa.htm: Torquemada, il validatore per l'accessibilità in lingua italiana

http://www.pubbliaccesso.gov.it/: Centro Nazionale per l'Informatica nella Pubblica Amministrazione

http://www.w3.org/WAI/: sito del Working Accessibility Initiative del W3C

http://www.w3.org/TR/WCAG10/: Web Content Accessibility Guidelines 1.0, Raccomandazione del W3C per l'accessibilità dei siti web

http://www.w3.org/TR/WCAG20/: Web Content Accessibility Guidelines 2.0, documento del 30 giugno 2005, ancora in fase di revisione

http://www.w3.org/TR/WCAG10/full-checklist.html: lista di controllo delle Checkpoints per le Web Content Accessibility Guidelines 1.0

http://www.w3.org/TR/WCAG20/checklist.html: lista di controllo per Web Content Accessibility Guidelines 2.0, documento del 30 giugno 2005, ancora in fase di revisione

http://europa.eu.int/scadplus/leg/it/s21012.htm: la Società dell'Informazione nel sito della Comunità Europea

http://www.annoeuropeodisabili.it/default.html: sito ufficiale italiano dell'iniziativa "2003 anno europeo delle persone con disabilità"

http://webxact.watchfire.com/: WebXACT, validatore gratuito per l'accessibilità dei siti web, sviluppato dalla Watchfire

http://www.cynthiasays.com/: sito di Cynthia dove è possibile validare le pagine web per l'accessibilità